CS 461 - CS Senior Capstone
Fall 2017
Software Requirements Specification

Connor I. Christensen
chriconn@oregonstate.edu
Lily M. Shellhammer
shellhal@oregonstate.edu
William B. Buffum
buffumw@oregonstate.edu

November 4, 2017

Abstract

Ninkasi Brewing Company is using an error prone, labor intensive and outdated brewing operations management system (BOMS) to manage the brewery data produced during the fermentation process. Our team (BrewHops) is tasked with creating a new system as both a proof of concept about new technology, and to give them another option for storing and viewing the brewery data. The new application will be web-based, to be lightweight, easy to use and accessible on many devices. The data will be hosted in a relational database to provide stability, power and consistent usability through time.

Keywords: Brewing, Operations, Management, Ninkasi

CONTENTS

1 INTRODUCTION

1.1 Purpose

This document will outline the project that satisfies the requirements for Ninkasis new brewing operations data management system. This includes a detailed description of the product created and the technology utilized to build the product. This document is intended for a technical audience, primarily for our instructors, Kirsten Winters and Kevin McGrath, and our sponsor, Daniel Sharp.

1.2 Scope

We will develop a web application and relational database model to manage Ninkasi data. We will perform a cost-benefit analysis to recommend hosting system with web service provider versus hosting on Ninkasi servers. Database will store:

- Beer type identifier [1]
- Batch number
- Generation (GEN) [2]
- Dry Hop/Adjunct Type [3]
- Hop amount
- Data on internal brewing vat (IBV) measurements

IBV data points include:

- Specific Gravity (S.G.) [4]
- pH level
- Alcohol By Volume (ABV)
- Beer temperature

Provided interface will allow brewers to manually insert new batch records, update incorrect data, and view existing data. Interface will allow Ninkasi personnel to view data using downloadable .csv files. We will gage adequacy of interface based on 60% approval from Ninkasi personnel, of which Daniel Sharp must approve. The goal of these requirements is to stop paper tracking of information and stop use of Excel as primary data storage.

Stretch goals include: login system and data viewing in browser (no download required).

We are not be responsible for advanced application security or direct control of any brewing process.

[1] A unique identifier for the type of beer, generally the name of the beer

[2] This the generation of the yeast we use to ferment the beer. We have to track this because we limit our yeast to 10 generations before repropagating it.

[3] This is a field that calls out the amount of hops that need to be added to the fermenter or adjunct (special ingredient) that needs to be added to the brite tank (like vanilla beans) It has as a status callout to track when the hops/adjuncts were added and by whom.

[4] A measure that represents the amount of sugar left in the beer. This number starts high and decreases throughtout fermentation. When it stops we move on to the next process step.

1.3 Definitions, acronyms and abbreviations

Data collected by our database will be referred to as cellaring data. This means data that has been collect while cellaring beers. Data points include but are not limited to: beer temperature, maturation time, amount of hops and other materials added.

BrewHops - The name of our team

BOMS - Brewery Operations Management System

1.4 References

No references as of 10/27/2017.

1.5 Overview

In the next sections, this document outlines details on the hardware, software, and user interface requirements for our project. It explains what needs to be done in order to launch our minimum viable product. Security, functionality, and organization are explained in detail.

2 Description

2.1 Product Perspective

2.1.1 System interfaces

Brewers will interact with our web app in the cellar. The web app interacts with the database and will enter data points from the brewer's entry into the form. If we meet our stretch goals, brewers will be able to extract Excel spreadsheets from our web app.

2.1.2 User interfaces

The user will interact with the web app through either an internet connected phone, tablet, desktop or laptop computer. Phones will operate almost exclusively in portrait mode. Apart from the occasional chart or data display, most of the use for the phone will be entering data, which benefits from a portrait layout. Tablets have the ability to use both portrait or landscape mode, depending on how they would like to use the device, and which method is more convenient for them. Desktops and laptops will work in a landscape mode. For big enough screens, extra space can be used to fit more functionality on one page without having to hide menu items or actions, such as extracting data from the database.

All Ninkasi brewers use a company provided phone and most of the brewer's work will be done through that phone. It's important that the site be consistent between iOS and Android to increase reliability, allow brewers to help each other with problems regardless of operating system, and reduce the learning curve. The environment in which the brewers are working requires that the data entry system be clear, easy to see and have a forgiving interface. For any new hires, the system should be intuitive[5]. Some phones may have smaller screens or may be held at a distance, so it's important that the font be big enough, the contrast strong enough, and the input boxes take up as much space as they can so they are easy to tap and input is easy to see. Brewers may have water, grain dust or other materials on their hands that will

[5] 60% acceptance of interface required to be considered ”intuitive”

interfere with the touch capabilities of the device. Making the interface forgiving by making buttons, input fields and any other interactive components large will reduce error and frustration.

If the system should use terminology that the brewers are familiar with. Reports should be color coded to keep the screen from filling with text and to allow the brewers to access the data at a glance.

2.1.3 Hardware interfaces

Ninkasi has a small server in their headquarters. A cost-benefit analysis will determine if this server will satisfy the hardware requirements of the system. If it will not, we will provide recommendations for web service providers.

2.1.4 Software interfaces

The user will interact with data via a web application. The application will expose input forms to submit data and downloadable .csv files to view data. We will release the software in stages, starting with a database, then a basic web input form connected to the database, then our final minimum viable product will include downloadable .csv files.

2.1.5 Communications interfaces

Web application will not implement other interfaces.

2.1.6 Memory constraints

Currently Ninkasi has space on a single server in a back room. Ninkasi will quickly grow out of server space which would constrain how much data we can host there. If Ninkasi decides to host our database on a 3rd party cloud hosting site, the memory constraints will be those of how much space they buy.

2.1.7 Site adaptation requirements

Brewers will adapt to using our web form rather than continue implementing the system they previously used. They will enter data not through paper logs or emails, but only through the web app. Our web form will need to connect to the chosen database, which may be hosted on site on Ninkasis server.

2.2 Product functions

Product include development of relational database and development of web application to process and present database information. Brewers will input information to the database using a form on the website, and retrieve information from the database as a .csv file.

2.3 User characteristics

Brewers using the web app will have all levels of education, including PhD, MS, BS, AAS, and no college experience. Their technical expertise about web development and coding in general will be minimal, but they will understand how to use mobile applications, how to access websites, and how to enter data into forms.

2.4 Constraints

2.4.1 Regulatory policies

Because we are creating this software from the group up and there are no previous regulations for a system like this, this section is not applicable for the BrewHops team.

2.4.2 Hardware limitations

Ninkasi has a small server that we may utilize for the database. If we decide not to use a third party hosting site, then the small server means there will only be the amount of free space on the server available. This is a not a limitation for our project so much as a limitation for any possible continuation of our project from Ninkasi.

2.4.3 Interfaces to other applications

BrewHops application will integrate with Tanknet. [6] The web app will automatically collect temperature information from Tanknet and store it in our database.

2.4.4 Parallel operation

Our web app will allow concurrent read access and use transactions for insertions and updates.

2.4.5 Audit functions

Not applicable for BrewHops team.

2.4.6 Control functions

Not applicable for BrewHops team.

2.4.7 Higher-order language requirements

Objects and callback functionality required.

2.4.8 Signal handshake protocols (e.g., XON-XOFF, ACK-NACK)

Not applicable for BrewHops team.

2.4.9 Reliability requirements

The web app will ensure data integrity for automated data collection. The web app will not ensure data integrity for manual data entry. Our site will be accessible without wait time longer than 5 seconds 90% of the time. If multiple users are using the system, it should not slow down operations on the web app by more than 3 seconds.

2.4.10 Criticality of the application

The web app is not critical to Ninkasi Brewing Companys success, but it would save the company data entry time. Ninkasi already has a working system that is not online. Our system would improve the current system by reducing time wasted from entry error and replace their working data entry processes.

2.4.11 Safety and security considerations.

This product will be light on security, as it is an internal application and should only be accessed by verified employees in the company. As this is not a public facing product, malicious activity is low on our list of priorities. Further details are outlined under the Specific Requirements section.

[6] Tanknet is a a hardware and software system that performs physical measurements inside brewing vats and automatically collects the data and exports it digitally

2.5 Assumptions and dependencies

BrewHops team will have access to Ninkasi Brewing Company's server to deploy application.

3 Specific requirements

3.1 External interfaces

Included in product are as follows:

- Database: A database will either be hosted on Ninkasi's server or through a 3rd party site. The input to the database will include multiple columns of cellaring data, who entered the information, and timestamps.
- Web Application: The web application will consist of a website that scales/changes functionality to fit the device.
  - Mobile Phone Page: The web app will scale to fit the mobile phone, mostly in portrait mode. The mobile version will be used for data entry and will have minimal options in comparison with the tablet or desktop. Data will not be extracted in .csv files through the mobile site.
  - Tablet Page: Tablets will operate in either portrait or landscape mode and will have the form and possibly more options for data extraction.
  - Desktop/Laptop Computer Page: This page will operate in landscape mode and will have the most options of the three pages.
  - Streatch goals include the ability to extracted database info in the form of .csv files

3.2 Functions

When data is being input into the database, there will be basic checks on the data. Information being entered that is far outside the range of expected values will prompt the user to verify that the information is indeed correct.

Every input form will strip out all html special characters to make sure that no harmful code can be injected into the database.

Stretch goals include:

- Adding a login interface to verify the users before they begin using the system. There is a database that will keep track of which users have already been authenticated. Users that are not registered in the system will be asked to create an account [7]. When doing so, they will be given a unique ID[8].
- Including a time and ID stamp on the data entered in case the company needs to trace back an error to a particular time or person.

3.3 Performance requirements

Multiple users on the web app at once will be permitted. There few Ninkasi brewers who will be entering data and it will be rare to have two entering at the same time. We will update this part of the doc when we receive the spreadsheet format from Daniel and are able to specify what values go where. 95% of interactions with the web page should happen in less than 5 seconds.

[7] should the brewers be allowed to create an account themselves?

[8] do the brewers already have an ID?

3.4 Logical database requirements

- Information pulled from Master Excel Spreadsheet
  - **FV**: [9]
  - **BEER**: [10], [11],
  - **BATCH**: [12]
  - **Generation**: The generation of the yeast we use to ferment the beer.
  - **Vol To Ferm (BBL)**:
  - **Vol To Bright (BBL)**:
  - **BRIGHT**: [13]
  - **% Yield**: [14]
  - **Dry Hop/Amount Type**: [15]
  - **Amount (lb)**: [16]
  - **Daily Monitoring Points**: [17], [18]
    - **Specific Gravity**: A measure that represents the amount of sugar left in the beer. This number starts high and decreases throughtout fermentation. When it stops we move on to the next process step
    - **pH**: pH level of batch in brewing vat.
    - **ABV**: Alcohol By Volumne of batch.
    - **Temp**: Temperature of batch.
    - **Action**: Action to be taken.
    - **By**: [19]
- Frequency of use

  Data currently updated daily and read throughout the day.
- Accessing Capabilities

  Database will be access from database access layer of web application.

3.5 Design constraints

Ninkasi currently has one server and a database hosted in microsoft access[20]. Our job is to create, or aid in creating, a new database system. We can either use their existing platforms, meaning limited space, or create a database on a third

[9] Is this the vat identifier?

[10] Should these be abbreviations of beer names, or is it preferable to store complete name?

[11] Are beer names associated with unique brewing processes

[12] What are the format specifications for textitbatch identifiers?

[13] What is this and when is it stored?

[14] Is this stored once per batch?

[15] Is this two distinct pieces of data that are shown together?

[16] Is this manually measured by brewers or automatically measured with sensors?

[17] How frequently should these data points be updated?

[18] If measured multiple times per day, should history of data be shown for individual day?

[19] Is this the person taking the action?

[20] is this an accurate?

party hosting site. Either option is limited by time, we can only implement so much of the data management throughout the timeline of this project. Stretch goals are to expand the database and data entry options, given we have time to do so.

3.5.1 Standards compliance

There are no standards compliance requirements for our software.

3.6 Software system attributes

3.6.1 Reliability

There is room for occasional bugs given that they do not damage data or inhibit the brewery process. Reliability of the system is a desirable goal, but the product is not a mission critical piece of software. Down time would be frustrating for employees, but would not render the company useless. That being said, its a reasonable goal that the system should be up 96% of the time[21].

3.6.2 Availability

The system should not require a reboot, recovery or restart. Stretch goals include making a daily backup of the information in the database if the system were to crash or the data were to be corrupted.

3.6.3 Security

This product will be light on security, as it is an internal application and should only be accessed by verified employees in the company. As this is not a public facing product, malicious activity is low on our list of priorities. Stretch goals will focus on security such as providing a login interface for employees, adding time stamps for data entered, and providing an easy method to find out who was responsible for entering incorrect data or misused the system. There will be some checks built into the delivered product, but those will primarily be designed for taking care of user error rather than malicious intent. [22]

3.6.4 Maintainability

This software is designed to be a proof of concept that having a system like this would be beneficial to Ninkasi's business development. Therefore, maintainability is not only good as a good coding practice, but it is one of the core requirements for the project that the code be maintainable and easy for others to understand.

There should be a clear separation between the frontend and the database connections. Our current plan involves using a REST API structure to ensure that if someone wanted to swap out the interface, or if someone needed a new connection to the database to be made, that a new port could be opened up and accessed from that space.

For the most part, code should be broken up into discrete functions that would allow for easy unit testing in the future. Though unit testing is not required as a goal for this project, the ability to attach some kind of testing framework will be useful for our code.

[21] What kind of uptime would be acceptable for the company?

[22] What kind of security would the company like to have?

3.6.5 Portability

This program is designed to be highly portable and the format of the product was selected specifically for that reason. The percentage of code and components with host-depenedent elements should be as small as possible, likely around 40%. Our site will be built using web technologies that should be able to operate on any common browser such as Firefox, Chrome and Edge. The most dependent code we will have will be the code running the database, and that will simply depend on the confguration that the server will have. If someone wanted to migrate the product over to a different server, they would only have to be sure that the underlying programming language like PHP, Rails some form of SQL or other tech is installed [23].

3.7 Organizing the specific requirements

We will build a user class based system. This will allow us to separate responsibilities and functionality so individuals only have access to their required tasks. In this system, we can design it hierarchically so that managers have access to all data manipulation privileges while brewers only have read or write privileges through structure forms.[24]

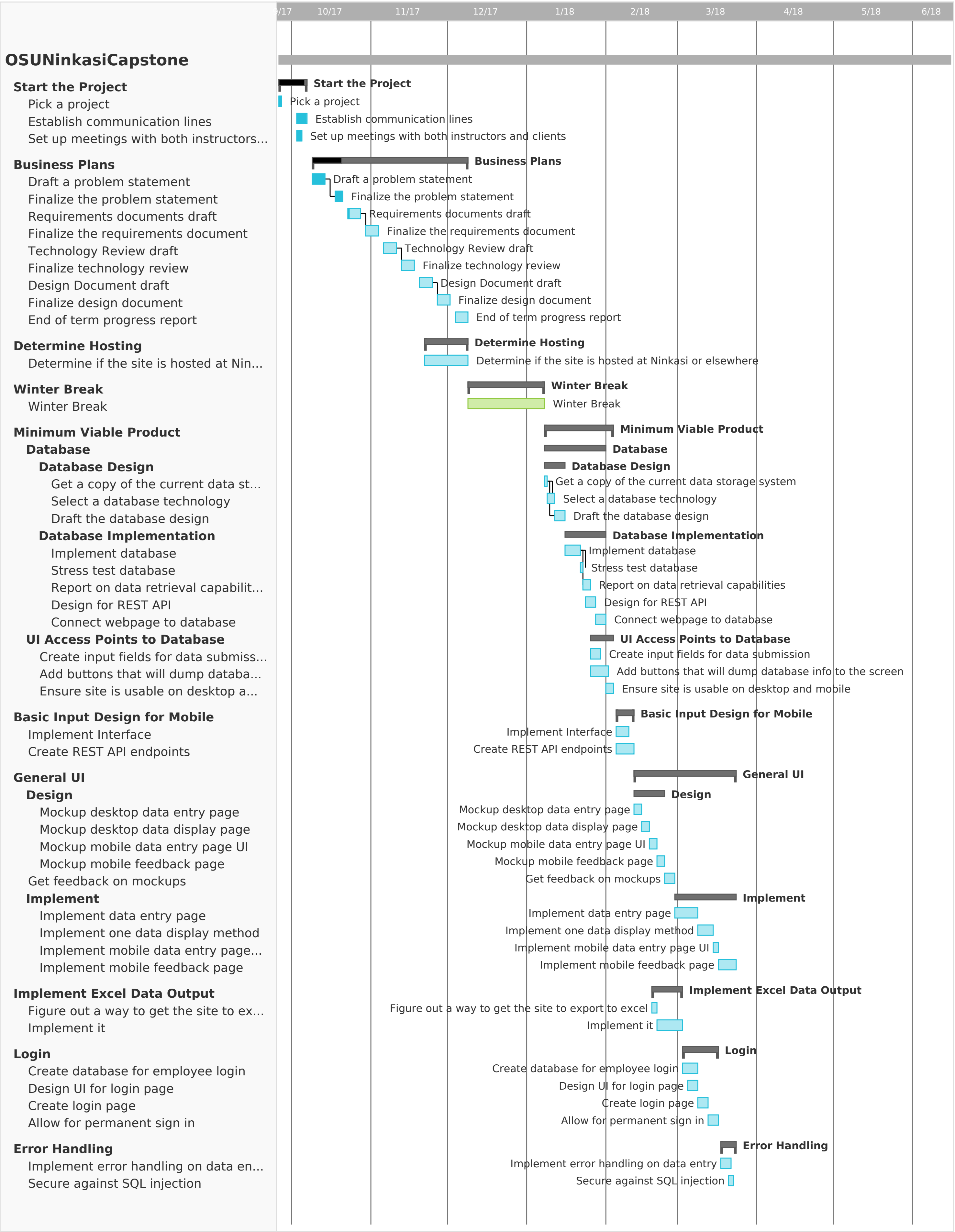
3.8 Comments

4 Appendixes

5 Index

[23] We will decide on the host technology later

[24] Is this level of separation necessary?



OSUNinkasiCapstone

Start the Project
- Pick a project
- Establish communication lines
- Set up meetings with both instructors...

Business Plans
- Draft a problem statement
- Finalize the problem statement
- Requirements documents draft
- Finalize the requirements document
- Technology Review draft
- Finalize technology review
- Design Document draft
- Finalize design document
- End of term progress report

Determine Hosting
- Determine if the site is hosted at Nin...

Winter Break
- Winter Break

Minimum Viable Product
- **Database**
  - **Database Design**
    - Get a copy of the current data st...
    - Select a database technology
    - Draft the database design
  - **Database Implementation**
    - Implement database
    - Stress test database
    - Report on data retrieval capabilit...
    - Design for REST API
    - Connect webpage to database
- **UI Access Points to Database**
  - Create input fields for data submiss...
  - Add buttons that will dump databa...
  - Ensure site is usable on desktop a...

Basic Input Design for Mobile
- Implement Interface
- Create REST API endpoints

General UI
- **Design**
  - Mockup desktop data entry page
  - Mockup desktop data display page
  - Mockup mobile data entry page UI
  - Mockup mobile feedback page
- Get feedback on mockups
- **Implement**
  - Implement data entry page
  - Implement one data display method
  - Implement mobile data entry page...
  - Implement mobile feedback page

Implement Excel Data Output
- Figure out a way to get the site to ex...
- Implement it

Login
- Create database for employee login
- Design UI for login page
- Create login page
- Allow for permanent sign in

Error Handling
- Implement error handling on data en...
- Secure against SQL injection

Check for other obvious security issu...

Spring Break

Spring Break

Auto Data Imports

Figure out possible technologies for i...

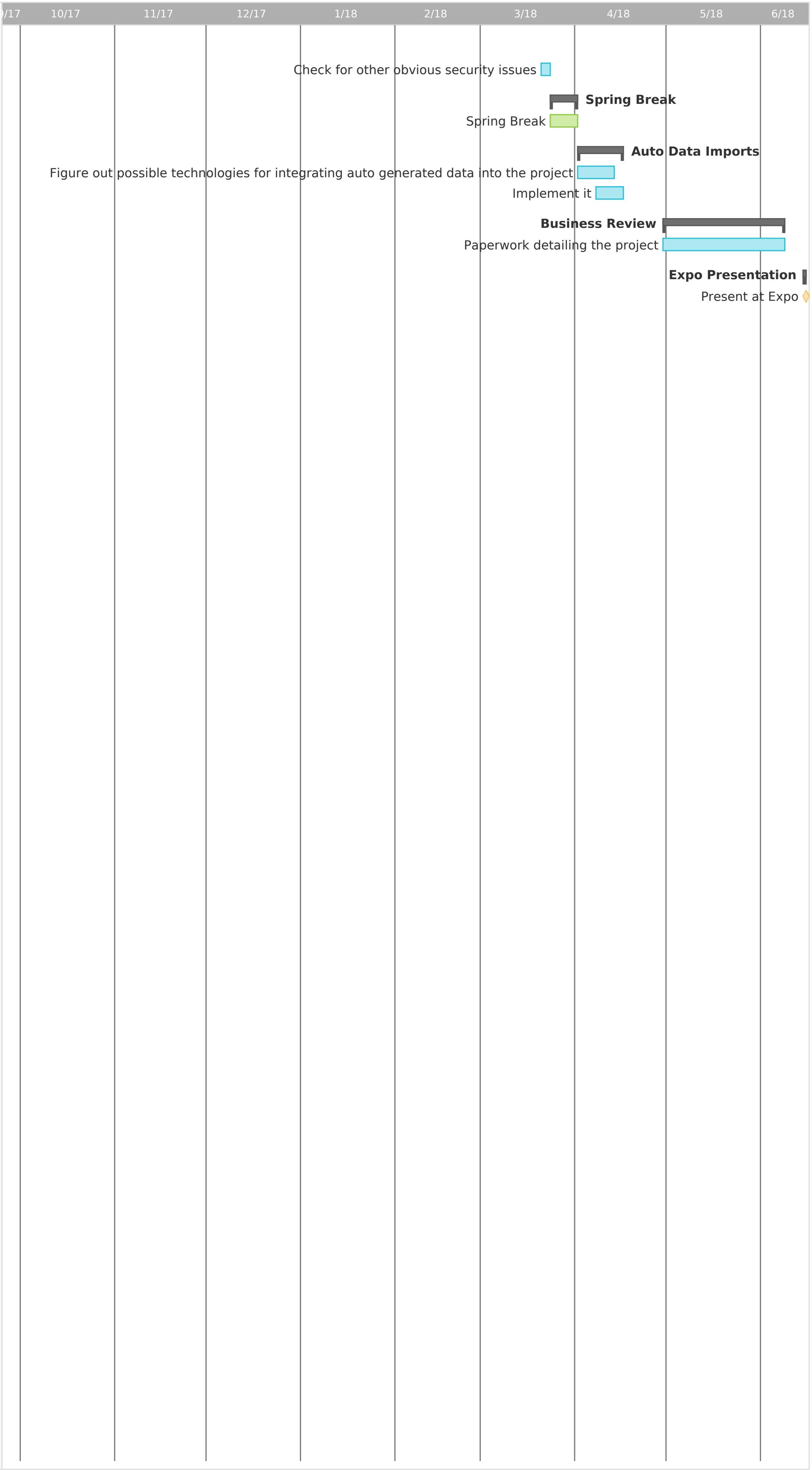
Implement it

Business Review

Paperwork detailing the project

Expo Presentation

Present at Expo


/17
10/17
11/17
12/17
1/18
2/18
3/18
4/18
5/18
6/18
Check for other obvious security issues
Spring Break
Spring Break
Auto Data Imports
Figure out possible technologies for integrating auto generated data into the project
Implement it
Business Review
Paperwork detailing the project
Expo Presentation
Present at Expo