Project Update: Winter

CS 462 Winter 2019

Group 19 BrewHops: Ninkasi Brewing, Automating the brewing process

Brennan Douglas
douglbre@oregonstate.edu
Dan Van Horn
vanhornd@oregonstate.edu
Henry Peterson
peterhen@oregonstate.edu
Bailey Singleton
singletb@oregonstate.edu

February 18, 2019

Abstract

This document outlines the progress that has been made in the Ninkasi Brewhops system's development including improvements to the development process, milestones that have been reached over the term, and specific improvements to the system itself. The problems the team has faced are described as well as the solutions that were implemented with the help of insightful client collaboration. The future development plans for the project are detailed according to the timeline and what the team has experienced during the development process thus far. Included are snippets of code that improved the project by utilizing the newest technology features and the maintainability of the codebase.

Contents

1 Recap

The Ninkasi Brewhops project is about bringing the brewing tracking process to the modern age. The web application is built with a front end that lets employees enter information about the current batch being brewed. It also has tracking features that provide history to the users, so they can see past batches in tanks and graphs that will give insight to the lifecycle of the batch. There will be a history of every change made to a batch or tank, and it will record who does it. They can export this information from the back-end database, that contains records of all the batches that have been entered. The files come out as Excel spreadsheets, which is how they used to keep track of the current processes. The tanks are laid out in an easy-to-understand manner, color coded in ways that represent the current action that is being applied to a tank. Coming from an Excel spreadsheet that was emailed across the company whenever a batch was updated, the Ninkasi Brewhops application will allow more efficient data entry and tracking, while simultaneously holding employees accountable for completing their tasks.

2 Progress

We fixed the application and API that we inherited from the previous team as the code did not run out of the box. Individually, the application and API ran (okay) with a little bit of tinkering. However, they did not communicate with each other as the API had different endpoints due to a redesign that happened after the previous school year. That took a large amount of effort and time to correct, but we got them working together allowing us to explore the application's current state.

Along with the re-factoring required to make the application and the API work together we also converted both from JavaScript into TypeScript. We started by applying the "any" catch all type everywhere we needed to and then began building up the real type that were being used around the application. This took a fair bit of time as well but has really benefited us as it creates a much smoother development process as we begin to tackle larger and more complex issues with the application.

To aid our development of the API we created a postman collection. This creates templates for the different url paths that can be hit and allows us to test them with false data. By configuring this tool we gain an efficient and easy way to assure that the API is working, and that the inputs are correctly validated.

Once both of these were set up and working we moved onto some of the usability issues that the application had. We have a concept of a *tank* (where the beer batches are brewed) that can have one active batch at a time. The page that displayed the tank's current information had a link labeled "add data" which linked back to the home page (how you navigate to the tank page) as there was a form on there that forced you to manually select the tank, the current recipe, and type in the correct batch name (of the current batch on the tank) to enter any data. All of this was redundant as it was already known on the specific tank's page, which was just navigated away from. So, we spent time redesigning this component to reduce the amount of information required for data entry and moved it to the individual tanks data entry page.

The life-cycle of a batch was also completed. Before the creation, updating, and completion of a batch associated with a tank was hidden from the user and attempted to "magically" happen all from the data entry component. This was confusing to the point where our client would create a new tank before he closed the batch for the existing tank. To solve this the existing data entry form was delegated to only update a batch, a new complete batch button was added to close it, and a new create batch form was made to replace the data entry form when there was no batch present on a tank. This cleared up how the client was supposed to manage the batch allowing him to use the application properly with ease.

The application also had no navigation options, you could mainly only go back to the previous page that your were on via the browser's back button. We added a navigation bar to provide links to all the root pages that were available to the user. This also included removing a button on the login page that was the only access to the admin portal for admin (it was required that you choose between that and the application itself). Now the admin can access that page from the navigation bar and the link only appears for them.

We also updated the batch history page to make it much more readable and to add graphs to it so that its data can still be visualized (even when it is not active in the tank). Further, we updated the possible actions that can be performed on a tank itself, they now come directly from the spreadsheet of data that we were provided (the data that will reside in the application). This was made even more visible by coloring the actions (again according to the spreadsheet) and applying their colors to the tanks on which they are active on the homepage.

Overall, we have been tracking our progress against the gantt chart and we have been staying on pace with our plan, even ahead in some areas. We already have alpha level functionality and we will most certainly have beta level functionality by the end of the term.

3 Plan

Many of our goals relating to the back-end and the user interface are either complete or currently in progress. That makes our plan for those area generally to fix bugs as they pop up and address changes that the client requests as we work through the iteration process. More and more issues will be found undoubtedly as the application is shown to the people who will be using it. These will ideally be made lighter by the implementation of testing. One user interface feature that we have been spotty on so far is mobile compatibility. For that, we will need to go through each of our pages, shrinking and stretching them to make sure that they are still accessible and make visual sense. This may require us to make changes so that the CSS or the Vue components are more adaptive or contain some more complex logic.

The next major aspects that will need to be given attention include testing and deployment. For testing, we still need to implement snapshot test so that we can ensure visual consistency on our pages. For this, we will need to go through each component and create the proper directory structures for the snapshots to be generated and stored in. Then, as those components get developed further, those tests will need to be updated with any intentional changes to the appearance of the app. In addition to snapshots, we will add unit test to our major components to ensure proper data flow through the application. We will create mock data to pass through the components and check that it is what is expected on the output.

For deployment, we still need to acquire some testing space so we can model the deployment process. Running on our local machines works well enough for regular development purposes, but an environment closer to one we actually plan to run in will allow us to really see if there are any issue with our setup and tear down scripts. Once that is in place and the application is in a more stable state with more client approval, we will need to acquire production server space. This will require some collaboration to make sure that it something that we can have access to and work with, then pass off to them once we are done working on the project. We will include supporting documentation so that anyone from outside our group who needs to restart, perform maintenance, or really do anything with the running instance is capable of doing so without too much issue.

4 Problems

At the beginning of development, we started with an initial code base from the previous Ninkasi Brewhops group. This gave us a good starting point, but with some pains. It is easier to write code, then it is to read and understand past code. We did not want to start from scratch, so we spent a good amount of time getting a project that did not run on arrival, to getting it at least running. From there, before we could develop new aspects of the application, we had to update the language from JavaScript to TypeScript. Getting the proper TypeScript loaders to work within WebPack took longer than we would have liked. But once the loaders worked, it cut down the development time.

Hosting has been another concern. To show our client our progress, we would like to be able to give him a link that would let him see where we are at. So far, the only way to give him updates is by screen sharing our project, and clicking around to show him what we have done. After talking with the school of EECS, they are unsure they can give us a spot on the server that can be accessed from outside the University grounds (without a VPN). So far we are looking into the hosting options suggested by the Tech Review, but none of these seem like they fit our bill, as we have progressed along with the project.

The largest issue, security-wise with our project was discovered on the login page. When checking if a user is in the system, they must enter a username and password. Upon receiving the credentials, the application would send the entire collection of passwords to the client side, encrypted. This is a huge security issue, as someone with the knowledge could take these passwords, and use them to decrypt their hidden information, thus granting them access to the web application. We have since fixed this by taking the encrypted password, and checking it with the back-end, and then returning if they are correct or not to the front-end, and letting the user in or not.

5 Interesting Code

Converting the project to Typescript gave us the opportunity to explicitly declare interfaces that declare function return types, parameters and object properties. This was instrumental when the team used these functions because the IDE would show an error if there was a type mismatch. Shown below are the interfaces for our Data Access layer, which defines all interactions with our database.

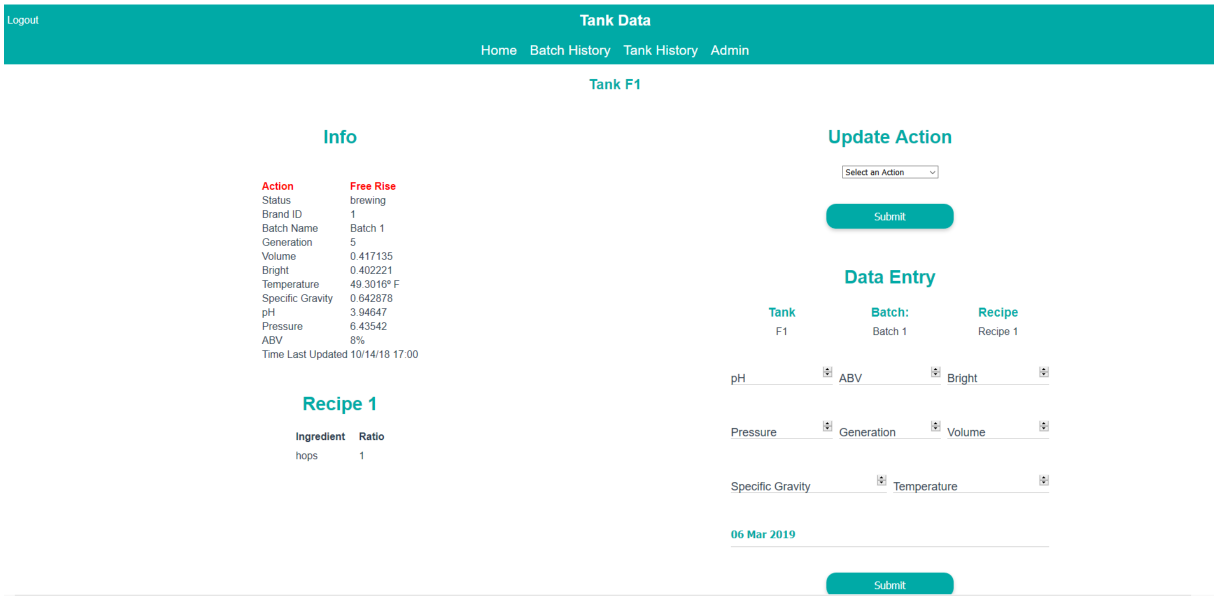
```
export interface ICrudController {
  tableName: () => string;
  connect: () => Promise<void>;
  disconnect: () => Promise<void>;
  create: (columns: any, conditions: any, escaped: any[]) => Promise<QueryResult>;
  createInTable: (
    columns: any,
    table: any,
    conditions: any,
    escaped: any[]
  ) => Promise<QueryResult>;
  read: (columns: string, conditions: string, escaped: any[]) => Promise<QueryResult>;
  readById: (escaped: any) => Promise<QueryResult>;
  readByUsername: (username: any) => Promise<QueryResult>;
  readInTable: (columns: any, table: any, conditions: any, escaped: any[]) => Promise<QueryResult>;
  update: (columns: any, conditions: any, escaped: any[]) => Promise<QueryResult>;
  // tslint:disable-next-line:no-reserved-keywords
  delete: (conditions: any, escaped: any[]) => Promise<QueryResult>;
  deleteById: (escaped: any[]) => Promise<QueryResult>;
  deleteInTable: (table: any, conditions: any, escaped: any[]) => Promise<QueryResult>;
}

export interface IPostgresController extends ICrudController {
  splitObjectKeyVals: (obj: any) => any;
  buildQueryByID: (key: string, value: string) => string;
  buildUpdateString: (keys: any) => any;
}
```

The following code is the refactored version of a very large chain of requests from the original application. It contains a lot of logic and even now, it takes a while to understand. The original code however, was nested six layers deep and was incredibly difficult to read. This code is objectively easier to reason about and understand.

```
try {
  const batchResponse = await this.$http.get(`${process.env.API}/batches`);
  const tanksResponse = await this.$http.get(`${process.env.API}/tanks`);
  const tasksResponse = await this.$http.get(`${process.env.API}/tasks`);
  const actionsResponse = await this.$http.get(`${process.env.API}/actions`);
  const recipeResponse = await this.$http.get(`${process.env.API}/recipes`);
  for (const tankInfo of tanksResponse.data) {
    // create a temporary tank for us to fill with data
    const tank: ITank = {
      // keep track of tank id for searching
      id: tankInfo.id,
      // keep track of tank name for displaying
      name: tankInfo.name,
      status: tankInfo.status
    };
    for (const batch of batchResponse.data) {
      // if our batches tankID matches our tankID
      if (batch.tank_id === tank.id) {
        tank.batch = {};
        // add in our batchesID to the tank info box
        tank.batch.id = batch.id;
        tank.batch.name = batch.name;
        // add the recipeID to the tank info box
        tank.recipe_id = batch.recipe_id;
      }
    }
    for (const recipeHistory of recipeResponse.data) {
      if (tank.recipe_id === recipeHistory.id) {
        tank.airplane_code = recipeHistory.airplane_code;
      }
    }
    if (tank.batch) {
      const versionsResponse = await this.$http.get(
        `${process.env.API}/versions/batch/${tank.batch.id}`
      );
      // keep track of most recent date with a starting low value
      let max = moment('1995-07-29');
      // for every data point we have in a batch
      for (const batchHistory of versionsResponse.data) {
        // if the date is the largest, it is the most recent one
        if (moment(batchHistory.updated_at) > max) {
          max = moment(batchHistory.updated_at);
          tank.pressure = batchHistory.pressure;
          tank.temperature = batchHistory.temperature;
        }
      }
      // find task associated with tank
      for (const task of tasksResponse.data) {
        if (tank.batch.id === task.batch_id) {
          // if task has our batch id
          tank.action_id = task.action_id; // save the asscoiated action
        }
      }
    }
    // find action associated with task
    for (const action of actionsResponse.data) {
      if (tank.action_id === action.id) {
        tank.action = action.name;
        tank.action_id = `action${action.id}`;
      }
    }
    // push data holder to the tanks array
    this.tanks.push(tank);
  }
  this.tanks.sort(this.sortTanks);
```

6 Application Screenshots



Tank F1

Info

Action	Free Rise
Status	brewing
Brand ID	1
Batch Name	Batch 1
Generation	5
Volume	0.417135
Bright	0.402221
Temperature	49.3016º F
Specific Gravity	0.642878
pH	3.94647
Pressure	6.43542
ABV	8%
Time Last Updated	10/14/18 17:00

Recipe 1

Ingredient	Ratio
hops	1

Data Entry

Tank	Batch:	Recipe
F1	Batch 1	Recipe 1

06 Mar 2019

Fig. 1. The new tank info page.

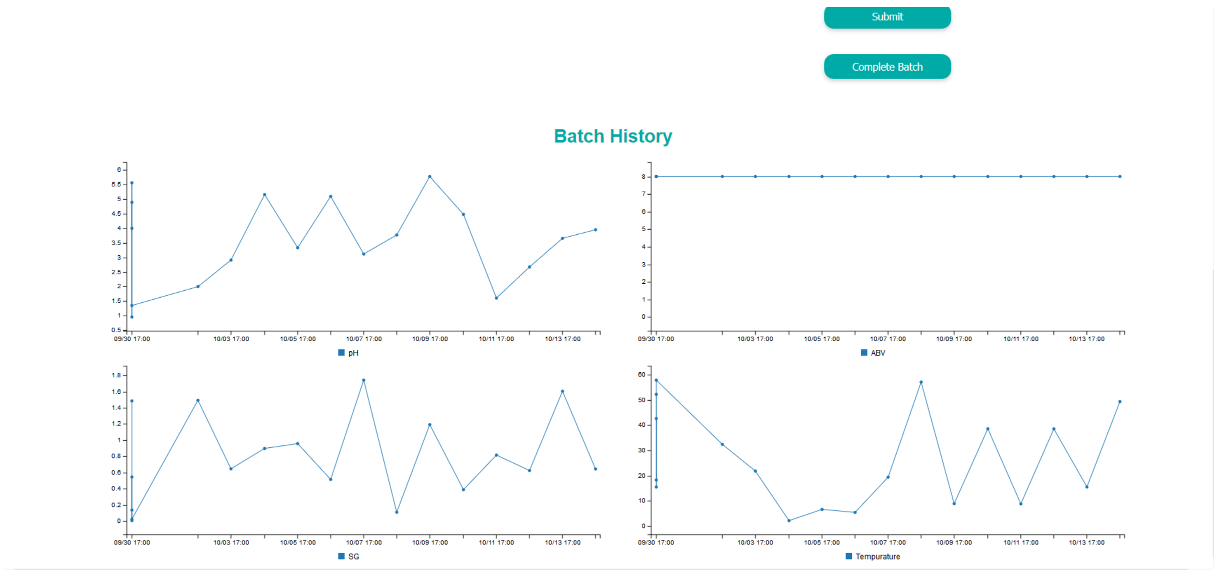
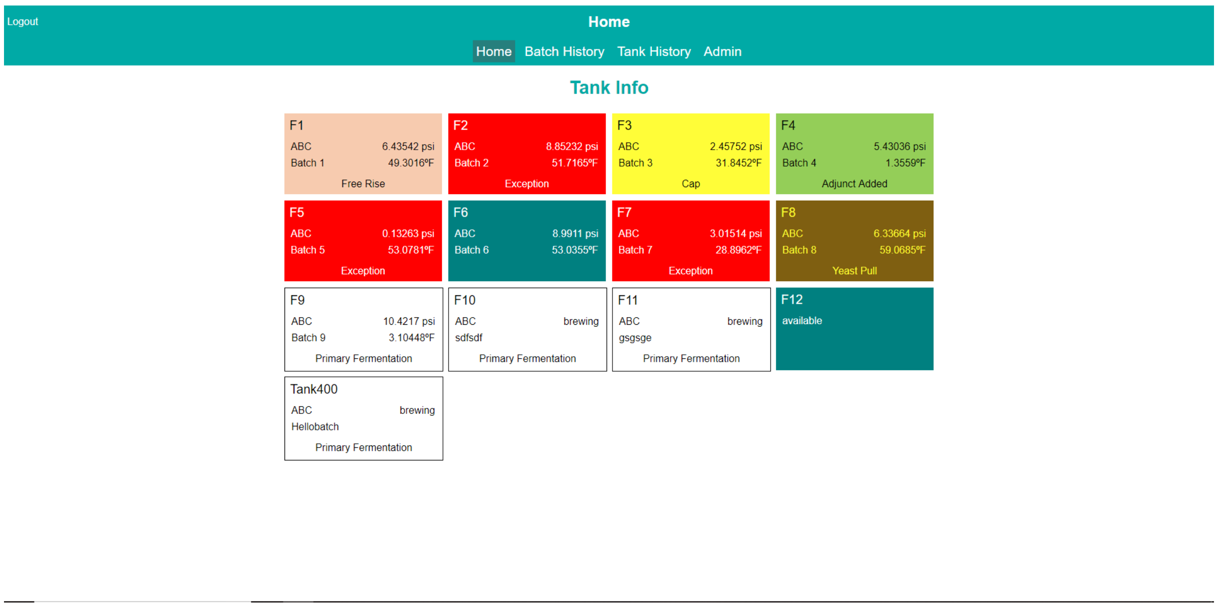


Fig. 2. The charts on the tank info page.



Logout

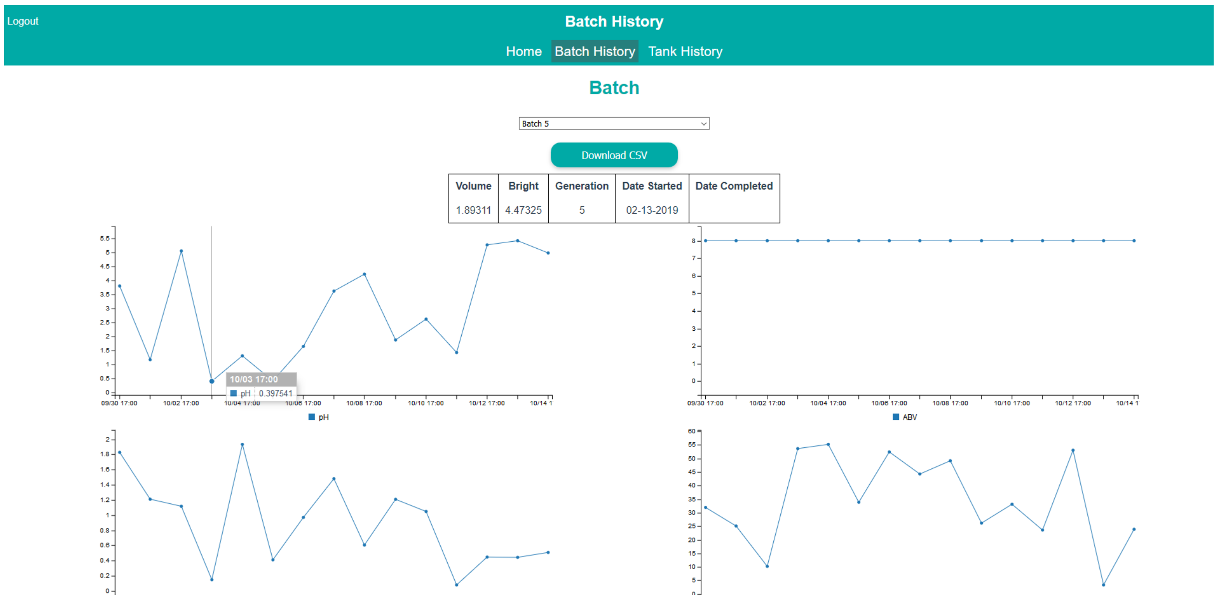
Home

Home Batch History Tank History Admin

Tank Info

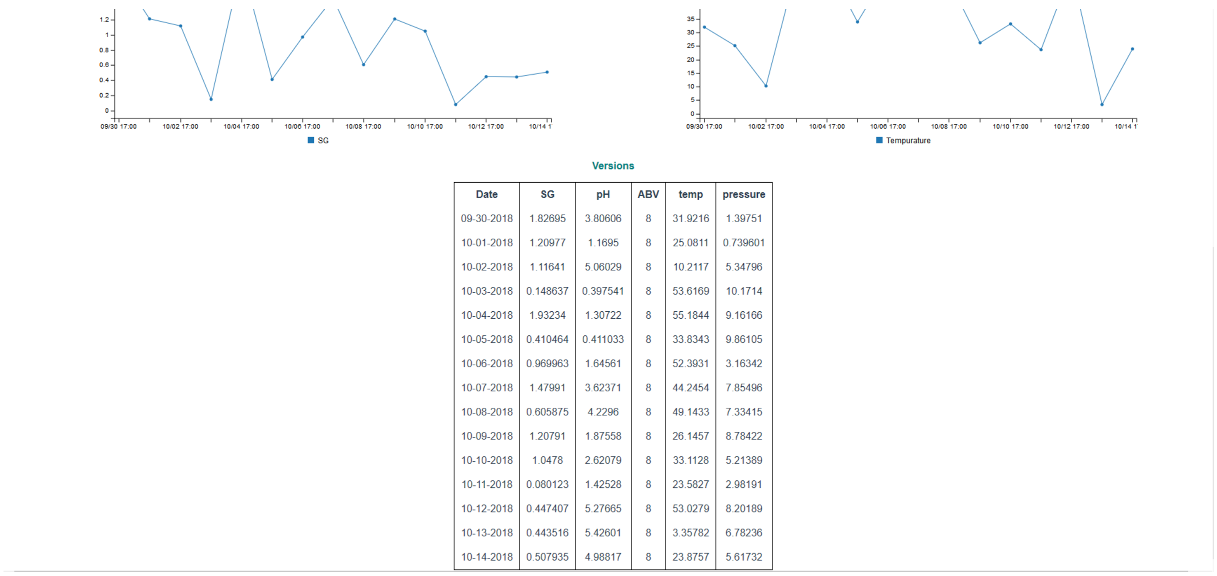
F1 ABC 6.43542 psi Batch 1 49.3016ºF Free Rise	F2 ABC 8.85232 psi Batch 2 51.7165ºF Exception	F3 ABC 2.45752 psi Batch 3 31.8452ºF Cap	F4 ABC 5.43036 psi Batch 4 1.3559ºF Adjunct Added
F5 ABC 0.13263 psi Batch 5 53.0781ºF Exception	F6 ABC 8.9911 psi Batch 6 53.0355ºF	F7 ABC 3.01514 psi Batch 7 28.8962ºF Exception	F8 ABC 6.33664 psi Batch 8 59.0685ºF Yeast Pull
F9 ABC 10.4217 psi Batch 9 3.10448ºF Primary Fermentation	F10 ABC brewing sdfsdf Primary Fermentation	F11 ABC brewing gsgsge Primary Fermentation	F12 available
Tank400 ABC brewing Hellobatch Primary Fermentation			

Fig. 3. Color coded tank monitoring by current action.



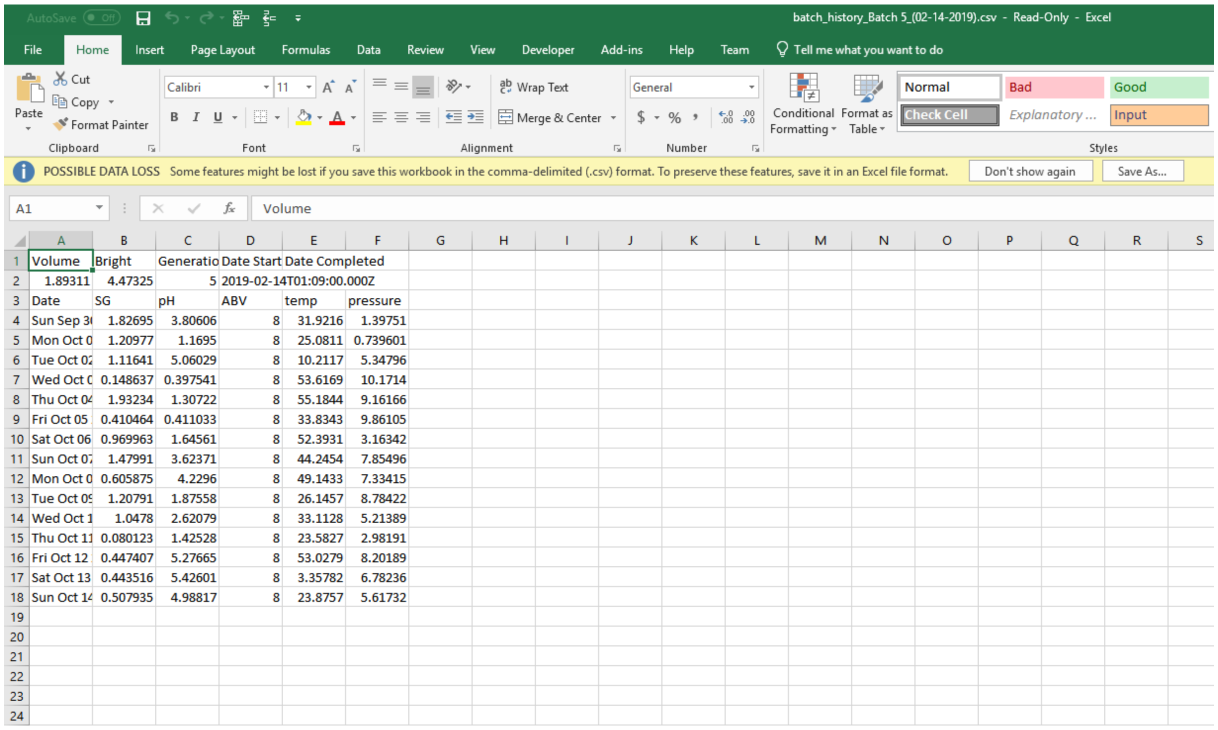
Volume	Bright	Generation	Date Started	Date Completed
1.89311	4.47325	5	02-13-2019	

Fig. 4. Updated batch history page (top).



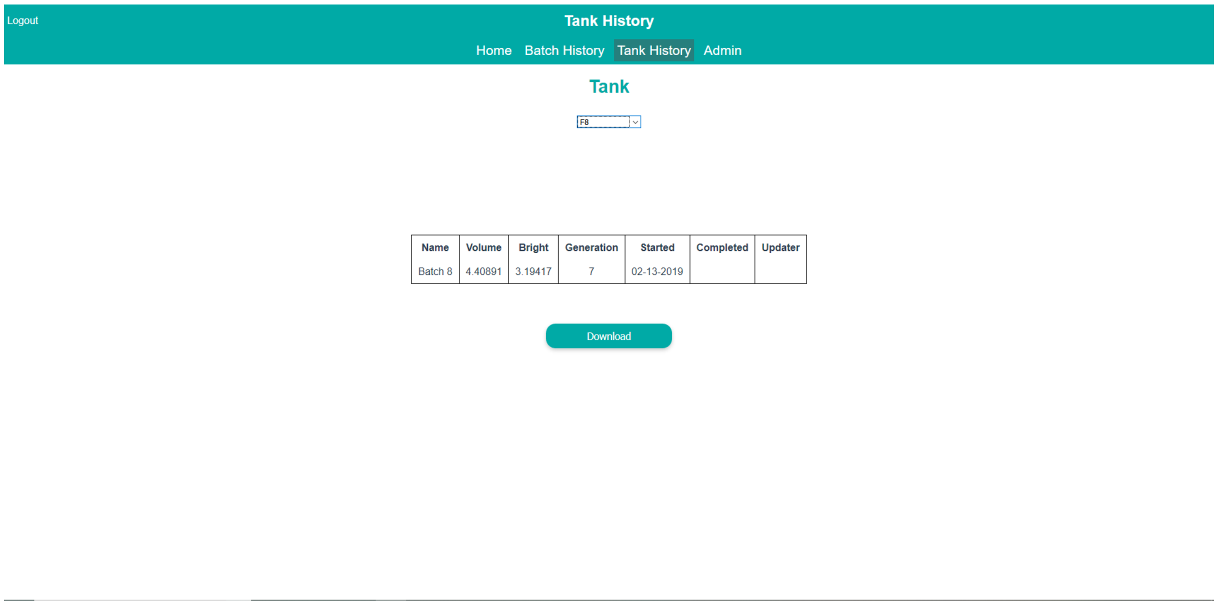
Date	SG	pH	ABV	temp	pressure
09-30-2018	1.82695	3.80606	8	31.9216	1.39751
10-01-2018	1.20977	1.1695	8	25.0811	0.739601
10-02-2018	1.11641	5.06029	8	10.2117	5.34796
10-03-2018	0.148637	0.397541	8	53.6169	10.1714
10-04-2018	1.93234	1.30722	8	55.1844	9.16166
10-05-2018	0.410464	0.411033	8	33.8343	9.86105
10-06-2018	0.969963	1.64561	8	52.3931	3.16342
10-07-2018	1.47991	3.62371	8	44.2454	7.85496
10-08-2018	0.605875	4.2296	8	49.1433	7.33415
10-09-2018	1.20791	1.87558	8	26.1457	8.78422
10-10-2018	1.0478	2.62079	8	33.1128	5.21389
10-11-2018	0.080123	1.42528	8	23.5827	2.98191
10-12-2018	0.447407	5.27665	8	53.0279	8.20189
10-13-2018	0.443516	5.42601	8	3.35782	6.78236
10-14-2018	0.507935	4.98817	8	23.8757	5.61732

Fig. 5. Updated batch history page (bottom).


AutoSave Off
batch_history_Batch 5_(02-14-2019).csv - Read-Only - Excel
File Home Insert Page Layout Formulas Data Review View Developer Add-ins Help Team Tell me what you want to do
Paste Cut Copy Format Painter Clipboard
Calibri 11 Font
Wrap Text Merge & Center Alignment
General Number
Conditional Formatting Format as Table
Normal Bad Good Check Cell Explanatory ... Input Styles
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.
Don't show again Save As...
A1 Volume

	A	B	C	D	E	F
1	Volume	Bright	Generatio	Date Start	Date Completed	
2	1.89311	4.47325	5	2019-02-14T01:09:00.000Z		
3	Date	SG	pH	ABV	temp	pressure
4	Sun Sep 3(	1.82695	3.80606	8	31.9216	1.39751
5	Mon Oct 0	1.20977	1.1695	8	25.0811	0.739601
6	Tue Oct 02	1.11641	5.06029	8	10.2117	5.34796
7	Wed Oct (	0.148637	0.397541	8	53.6169	10.1714
8	Thu Oct 04	1.93234	1.30722	8	55.1844	9.16166
9	Fri Oct 05	0.410464	0.411033	8	33.8343	9.86105
10	Sat Oct 06	0.969963	1.64561	8	52.3931	3.16342
11	Sun Oct 07	1.47991	3.62371	8	44.2454	7.85496
12	Mon Oct 0	0.605875	4.2296	8	49.1433	7.33415
13	Tue Oct 09	1.20791	1.87558	8	26.1457	8.78422
14	Wed Oct 1	1.0478	2.62079	8	33.1128	5.21389
15	Thu Oct 11	0.080123	1.42528	8	23.5827	2.98191
16	Fri Oct 12	0.447407	5.27665	8	53.0279	8.20189
17	Sat Oct 13	0.443516	5.42601	8	3.35782	6.78236
18	Sun Oct 14	0.507935	4.98817	8	23.8757	5.61732

Fig. 6. New batch history page CSV export.



Name	Volume	Bright	Generation	Started	Completed	Updater
Batch 8	4.40891	3.19417	7	02-13-2019		

Fig. 7. Updated tank history page.

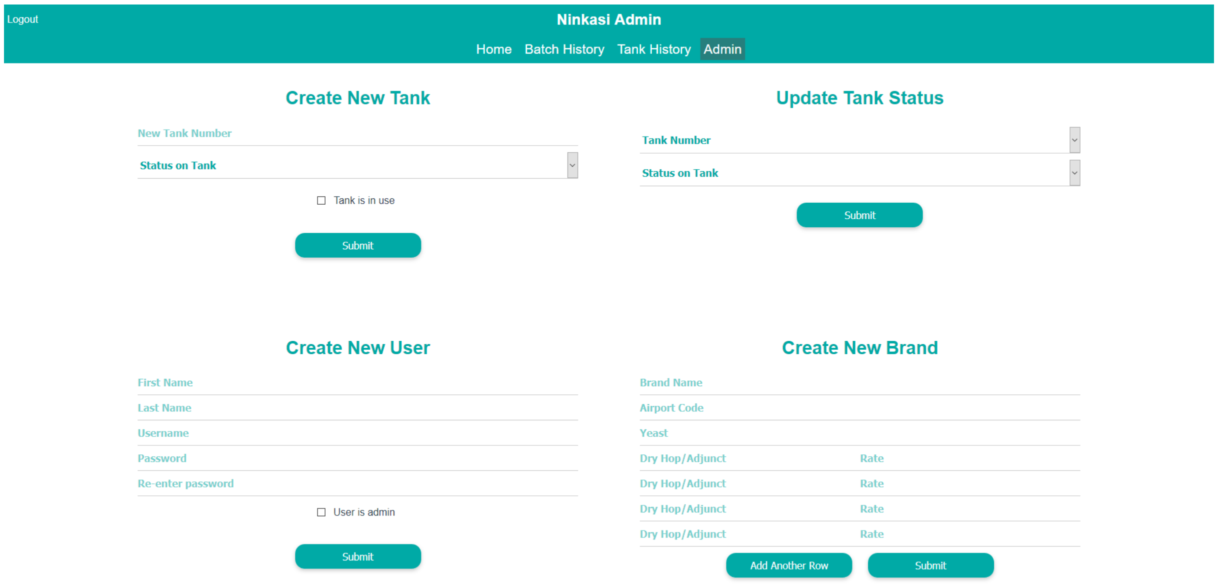


Fig. 8. Admin page.

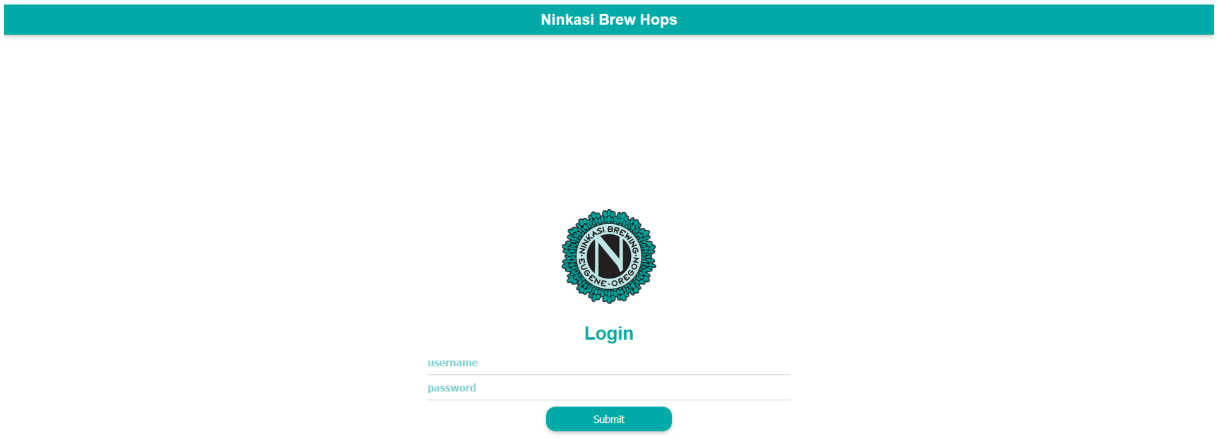


Fig. 9. Improved login page.

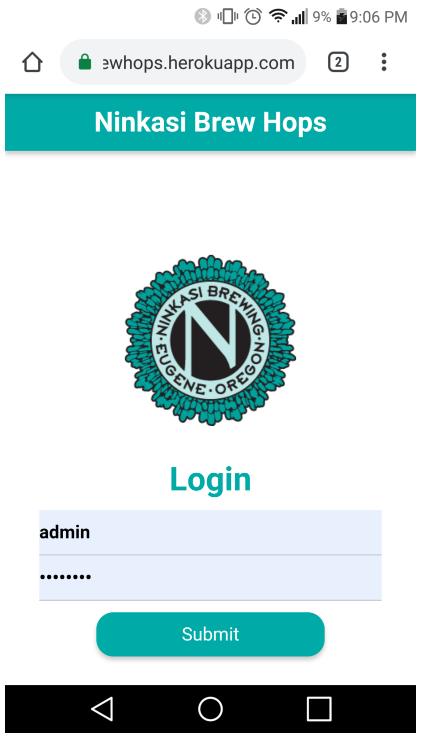
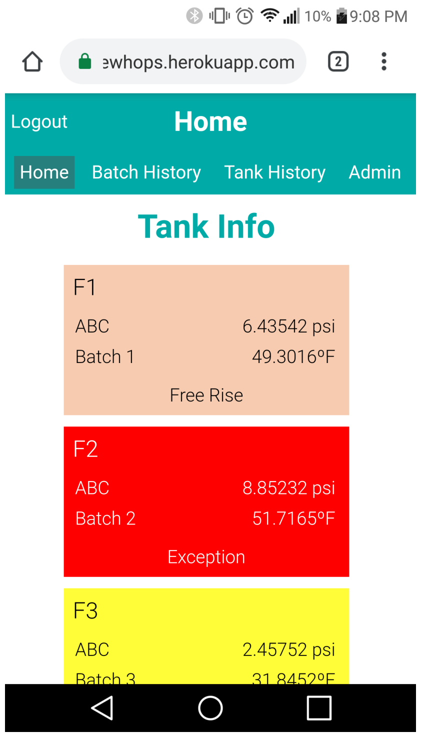


Fig. 10. Mobile login page.

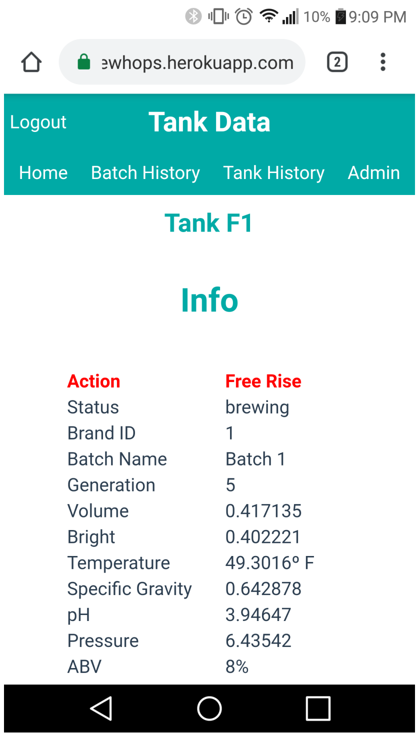


F1
ABC 6.43542 psi
Batch 1 49.3016ºF
Free Rise

F2
ABC 8.85232 psi
Batch 2 51.7165ºF
Exception

F3
ABC 2.45752 psi
Batch 3 31.8452ºF

Fig. 11. Mobile tank monitoring page.


9:09 PM

ewhops.herokuapp.com

Logout **Tank Data**

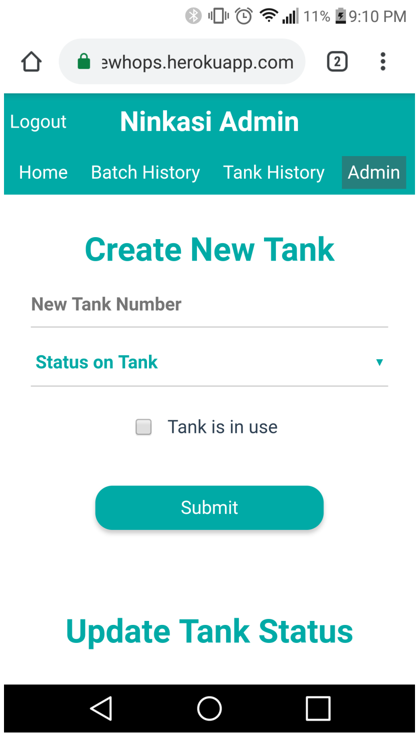
Home Batch History Tank History Admin

Tank F1

Info

Action	**Free Rise**
Status	brewing
Brand ID	1
Batch Name	Batch 1
Generation	5
Volume	0.417135
Bright	0.402221
Temperature	49.3016° F
Specific Gravity	0.642878
pH	3.94647
Pressure	6.43542
ABV	8%

Fig. 12. Mobile tank info page.


9:10 PM

ewhops.herokuapp.com

Logout **Ninkasi Admin**

Home Batch History Tank History Admin

Create New Tank

New Tank Number

Status on Tank

☐ Tank is in use

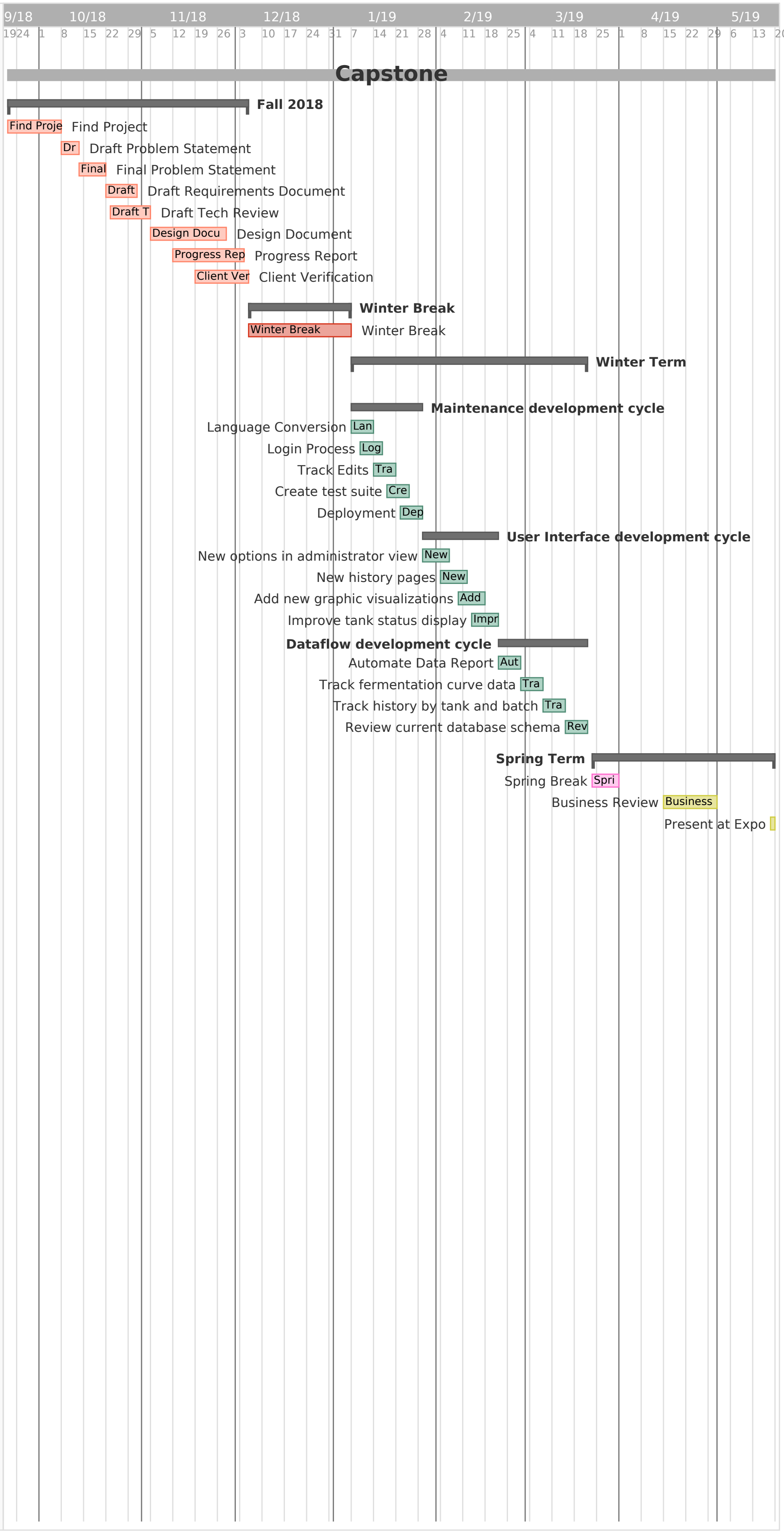
Submit

Update Tank Status

Fig. 13. Mobile admin page.

teamgantt
Created with Free Edition

Capstone	**0h**
Fall 2018	**0h**
Find Project	0
Draft Problem Statement	0
Final Problem Statement	0
Draft Requirements Document	0
Draft Tech Review	0
Design Document	0
Progress Report	0
Client Verification	0
Winter Break	**0h**
Winter Break	0
Winter Term	**0h**
Deployment	**0h**
Maintenance development cycle	**0h**
Language Conversion	0
Login Process	0
Track Edits	0
Create test suite	0
Deployment	0
User Interface development cycle	**0h**
New options in administrator view	0
New history pages	0
Add new graphic visualizations	0
Improve tank status display	0
Dataflow development cycle	**0h**
Automate Data Report	0
Track fermentation curve data	0
Track history by tank and batch	0
Review current database schema	0
Spring Term	**0h**
Spring Break	0
Business Review	0
Present at Expo	0


9/18
10/18
11/18
12/18
1/19
2/19
3/19
4/19
5/19
Capstone
Fall 2018
Winter Break
Winter Term
Maintenance development cycle
User Interface development cycle
Dataflow development cycle
Spring Term