Software Design Document

BeerTap

CMSC 495 Summer 2019

Team 3

Nishan Basnet

Steven Etienne

Kevin Hoy

Rebecca Lee

Hunter Smith

Contents

1. Change Log

Version	Primary Author	Description	Date Completed
draft	Hunter Smith	Created template and added initial input	6/13/2019
v.1	Kevin Hoy	Completed unfinished sections	6/16/2019
v.2	Rebbeca Lee	Completed read through and edits	6/16/2019
v.3	Hunter Smith	added phase I section, changed pics Completed A5. Human interface design	6/23/2019
v.4	Kevin Hoy	Updated phase I section to include test results	6/23/2019
v.5	Kevin Hoy	Updated phase II section to include test results	6/30/2019
v.6	Kevin Hoy	Updated phase III section to include test results	7/7/2019

2.

A. Software Design Description

A.1 Introduction

A.1.1 Purpose

The purpose of this design document is to provide a detailed description and illustration of the design elements for the BeerTap web application to allow for software development to proceed. This document will guide stakeholders in the applications design and implementation.

A.1.2 Scope

The scope of this project is the design of a web-based application called BeerTap. The application will allow users to discover details about beers and breweries, store their insights on beers they have tried through a simple rating system, share ratings with friends, and receive recommendations for new beers based on past ratings.

The target audience for BeerTap is beer enthusiasts who can leverage any device with a modern web browser and a desire to explore and participate with beer culture.

A.1.3 Overview

This design document will illustrate various design aspects of the BeerTap application including architecture, data/database design, and interface and component design.

A.1.4 References

IEEE Standard 1016−2009, IEEE Standard for Information Technology – Systems Design – Software Design Description, IEEE Computer Society, 2009.

API Documentation. *Untappd*, untappd.com/api/docs#start.

A.1.5 Definitions and Acronyms

Terms	Definitions
API	Application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. (Wikipedia)
PWA	Progressive web app (PWA), a browser-based application that has become an alternative to a native mobile app. (Techopedia)
BeerTap	A PWA that allows users to search for beers and breweries.
Webstack	A compilation of software applications, often needed for web development, especially for developing web applications and implementing websites.(Techopedia)

A.2 Overview

This design document is a written description of the BeerTap app for the software development team to refer to for guidance on the architecture of the project. The document is separated into four (4) sections; each containing one or more subsections as shown below:

- Software Design Description: Contains information about the purpose of the software, system architecture, data design, component design, and the human interface design.
- Software Plan: Contains information about the project schedule as well as roles and responsibilities.
- Test Plan: Contains information used to test the application such as test cases, test methodology, test deliverables, and resource needs.
- User Guide: Contains information to assist in using the applications major features and functions. For example: Navigating the screens; searching for beers; logging in; and sharing ratings.

A.3 System Architecture

A.3.1 Architectural Design

This project will use a Client-Server architecture. Web clients (browsers) will request services from servers (Web & DB) and will communicate using the Transport Layer Security (TLS) protocol. The integrity of the data requires serving the website via HTTPS and installing an SSL certificate.

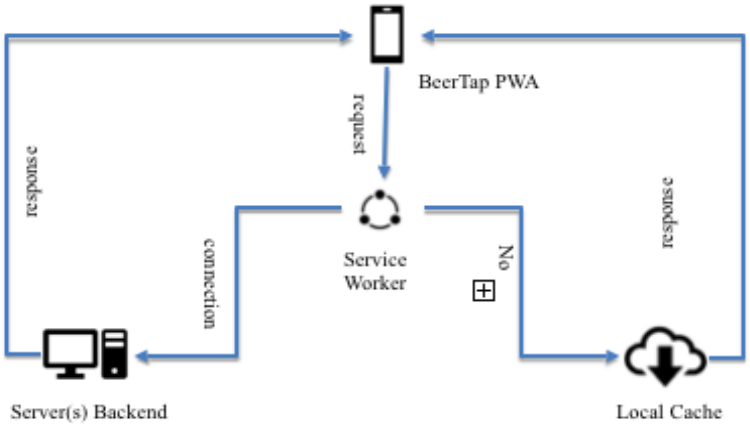

figure 3.1

A.3.1.1 Use Case

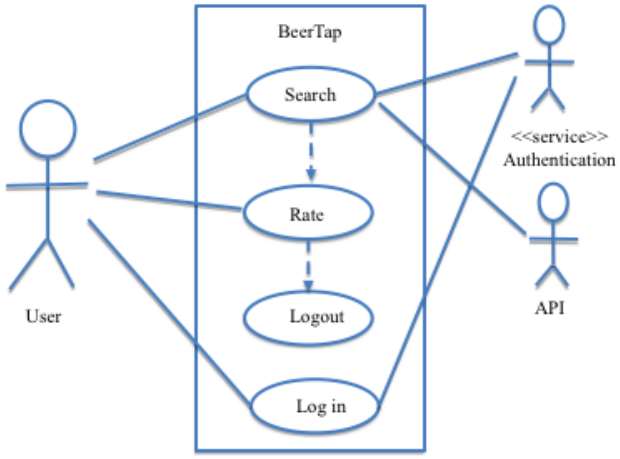

figure 3.2

A.3.2 Decomposition Description

The BeerTap PWA uses a webstack (React JavaScript, HTML5, and CSS3) to provide an interactive experience to the user. The user can interact with the application using either a computer or a mobile device. Static content is separated from dynamic content using the application shell. The shell is static while the content displayed in the shell is dynamic.

The Service Worker is a technical element that supports the main feature of a PWA – the ability to use the app offline. The service worker is a JavaScript file that runs separately from the app and responds to user interactions.

The Local Cache is used by the Service Worker to cache the application shell (interface) so that it loads instantly each time the app is loaded.

The backend servers include the Cloud based web server, the MySQL database server, and the Beer API server.

A.3.3 Design Rationale

The design was chosen for the following reasons:

- Because progressive web apps generally result in development savings in both time and effort compared to native apps. There is no need to develop two separate apps based on mobile OS.

- PWAs are not distributed through an app store. They are discoverable using a search engine.

- PWAs are easier to update and do not require developers to support older versions that have not been updated.

A.4 Data Design

A.4.1 Data Description

The BeerTap app will use a MySQL database that contains the following tables:

- Users
  - UserID INT(999) PK
  - Name VARCHAR(50)
  - City VARCHAR(50)
  - Password VARCHAR(15)
- Ratings
  - UserID INT(999) FK
  - BeerID INT(999) FK
  - Aroma INT(2)
  - Appearance INT(2)
  - Flavor INT(2)
  - Mouthfeel INT(2)
  - Overall INT(2)
  - Comments VARCHAR(255)

In addition, BeerTap will use a connection to the Beer data API using a unique key for authentication. The data elements of the API are:

- BeerID
- Name
- Maker
- Details
- Style

A.4.2 Data Dictionary

Item	Description
BeerID	This is the unique ID number for each beer item in the API
City	City where the user resides
Comment	Free form field that the user can fill out to provide additional details about the rated beer
Name	Name of the user
Overall	Rating of all beer attributes rated by the user
UserID	Unique ID number used to identify each user

A.5 Human Interface Design

A.5.1 Overview of User Interface

The user Interface is provided by a dynamic webpage in the user's device's browser. The interface is designed to scale for desktop and mobile devices providing the user a similar experience on either platform, although mobile devices are the primary intended platform. To accommodate mobile platforms the interface is responsive to touch, if enabled by the device, and feature large buttons with an attempt to minimize the amount of text.

A.5.2 Screen Images

A.5.2.1 Homepage

figure 5.1

Beertap's Homepage is the primary mode of user interaction. From the home page users can toggle the navigation bar at the top to search for beers, breweries, login, and overall navigate to different parts of the site.

A.5.2.2 Search Results Page

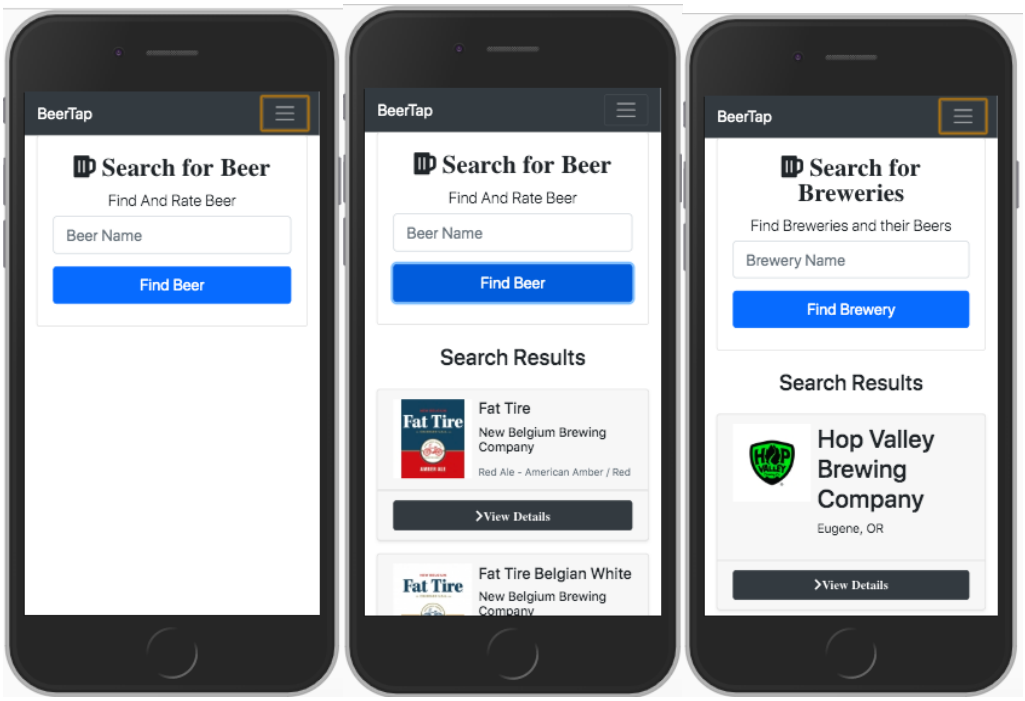

figure 5.2

The brewery and Beer Search pages provide a similar layout where the results are shown under the search window as illustrated by figure 5.2. When a search returns more than one result users can continuously scroll down to the bottom of the results

A.5.2.3 Searched Item's Details Page

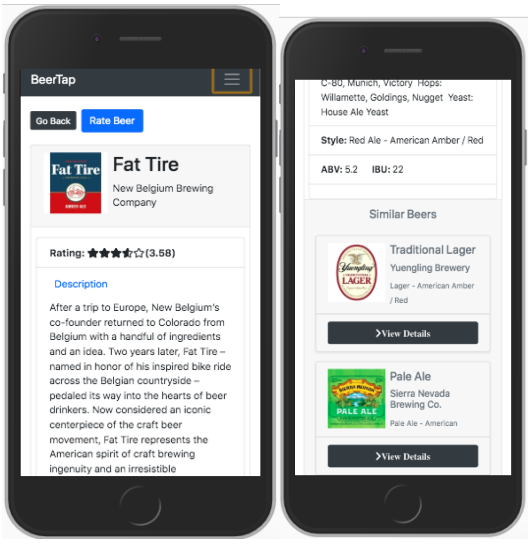

figure 5.3

As mentioned in Section A.5.2.2, whether the user wants to view beer or brewery details he/she can access the details by clicking the view details button of the desired item. In the case of beer details the user can rate the beer from this page by clicking the rate beer button (if logged in). Additionally, on the results page users can see similar beers by scrolling down.

A.5.2.4 Beer Rating Page

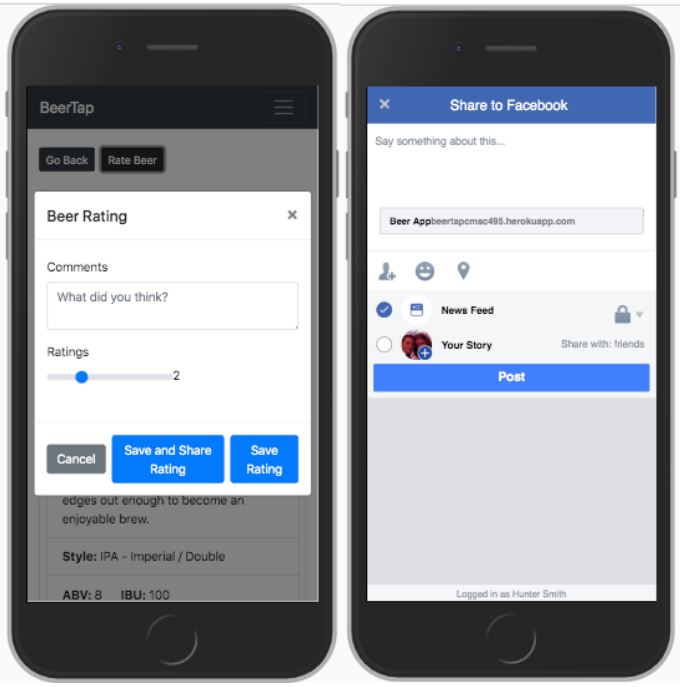

figure 5.4

The beer rating page provides the user with the ability to enter their rating of the selected beer and gives them the option to either save the rating or save and share the rating through social media by clicking the corresponding button. However, the beer rating functionality is only an available option when a user is logged into the website and multiple rating bars have yet to be added.

B. Software Plan

B.1 Overview

B.1.1 Schedule

The project should be completed and submitted by the end of the course(14July2019).

Date	Milestone	Description	Assigned
6/9/2019	Test Plan	Outline the test strategy, testing objectives, resources required, test schedule, and test deliverables.	Rebecca; Nishan
6/19/2019	Project Design	Document the design elements of the project: • Key Features • Structure • Criteria for success • Major Deliverables • Alternative Designs	Kevin
6/23/2019	Phase 1 Dev	Development of first version of the app. Exercise Test Plan to identify issues and areas for improvement	Steven (Backend); Hunter (Frontend); Rest of Team (Testing)
6/30/2019	Phase 2Dev	Development of second version of the app to address issues identified during the exercising of the test plan from Phase 1. Exercise test plan again.	Steven (Backend); Hunter (Frontend); Rest of Team (Testing)

7/7/2019	Phase 3 Dev	Development of third (and final) version of the app that has addressed issues identified in version 2. Exercise final phase of test plan to validate that the criteria for success in the project design has been achieved.	Steven (Backend); Hunter (Frontend); Rest of Team (Testing)

B.2 Roles and Responsibilities

Members	Roles
Kevin	Project Manager, Frontend Developer
Hunter	Lead Frontend Developer
Steven	Lead Backend Developer
Rebbeca	Documentation, Coder, Testing
Nishan	Documentation, Coder, Testing

C. Test Plan

C.1 Introduction

This test approach document describes the appropriate strategies, process, workflows and methodologies used to plan, organize, execute and manage testing of software projects

C.1.1 Scope

In Scope: The scope of the project is to develop a web-based application that can be used by any device with a modern web browser. Users of mobile devices will also have the ability/option to install the application to the device (Progressive Web Application).

The target audience of the application is beer drinkers who want a tool that will allow them to locate new beers and rate the beers that they have tasted.

Out-Of-Scope: There are various parts of applications that does not require testing. Some of them are:

a) Performance [Optional]

b) Cross browser [Optional]

C.1.2 Quality Objective

Comprised of three objectives,

c) Ensure the Application Under Test conforms to functional and non-functional requirements

d) Ensure the AUT meets the quality specifications defined by the client

e) Bugs/issues are identified and fixed before going live

C.1.3 Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like

a) QA Analyst: Tests all the required functionalities and performs required testing. (Black box)

f) Test Manager: Manages the team and makes sure documents are up to date and are distributed across the team. Monitor and manage testing integrity and Support testing activities.

C.1.4 Test Cases

C.1.4.1 Allow user to create account.

TC.N	Action	Expected Result
1	User opens the app	The system displays the app
2	User clicks on login	The system displays the screen giving the user the option to “Sign In” or “Join Now”.
3	User clicks on the Join Now button	The System displays the registration page to collect user data.
4	User fills out registration form and clicks on Create Account button	The System creates the user account and displays the main landing page of the app
5	User clicks on login button	The system authenticates the user based on information entered. If authentication is successful, the user will be shown a personalized welcome message.
6	User clock on the logout button	The system logs the user out and returns to the main landing page of the app.

C.1.4.2 Allow user to search beers

TC.N	Action	Expected Result
1	User opens the app	The system displays the app
2	User clicks on login	The system displays screen giving the user the option to “Sign In” or “Join Now”
3	User click on “Sign In” and provides account and password.	The system authenticates the user. If successful, the user will be presented with a personalized welcome message.
4	User searches the beer using the search bar	The System displays corresponding list of results related to a beer or beers in the area
5	User clicks on the beer	The system displays beer image with descriptions.
7	User clicks on log out button	The system displays sign in page

C.1.4.3 Allow user to rank beers.

TC.N	Action	Expected Result
1	User opens the app	The system displays the app
2	User clicks on login	The system displays login fields with username and password (*) as mandatory.
3	User enters username, passwords and clicks sign in button	The System displays homepage of the app System responds with incorrect login info if username/password are incorrect
4	User searches the beer using the search bar	The System displays corresponding list of results related to a beer or beers in the area
5	User clicks on the beer	The system displays beer image with descriptions.
6	User ranks the beer	The system saves the rank provided by user and displays it.
7	User clicks on log out button	The system displays sign in page

C.1.4.4 Get recommendations based on previous beer ratings

TC.N	Action	Expected Result
1	User opens the app	The system displays the app
2	User clicks on login	The system displays login fields with username and password (*) as mandatory.
3	User enters username, passwords and clicks sign in button	The System displays homepage of the app System responds with incorrect login info if username/password are incorrect
4	User shall be able to see recommended beers in his portal	The System works as expected
5	User clicks on log out button.	The system displays sign in page

C.1.4.5 Share beer information

TC.N	Action	Expected Result
1	User opens the app	The system displays the app
2	User clicks on login	The system displays login fields with username and password (*) as mandatory.
3	User enters username, passwords and clicks sign in button	The System displays homepage of the app System responds with incorrect login info if username/password are incorrect
4	User clicks on share icon	The system prompts for username to share within the app
5	User searches for another user who he/she wants to share	The System populates username based on user search criteria.
6	User shares the beer with friends through email	The system works as expected and email is sent and received
7	User clicks on log out button.	The system displays sign in page

C.1.5 Assumptions for test execution

For User Acceptance testing, the Developer team has completed unit, system and integration testing and met all the Requirements (including quality requirements) based on Requirement.

User Acceptance testing will be conducted by End-users

Use cases are approved by test lead within the team.

Test scripts are developed and approved.

Test Team will support and provide appropriate guidance to Developers to conduct testing.

Major dependencies should be reported immediately after the testing kickoff meeting.

C.1.6 Constraints for test execution

Developer will receive consolidated list of requests for test environment set up, user accounts set up, data set (actual and mock data), defect list, etc.

Developer will support ongoing testing activities based on priorities.

C.2 Test Methodology

C.2.1 Overview

Agile.

C.2.2 Test Levels

Test Levels define the Types of Testing to be executed on the Application Under Test (AUT). The Testing Levels primarily depends on the scope of the project, time and budget constraints.

Testing of an application can be broken down into three primary categories and several sub-levels. The three primary categories include tests conducted every build (Build Tests), tests conducted every major milestone (Milestone Tests), and tests conducted at least once every project release cycle (Release Tests). The test categories and test levels are defined below:

C.2.2.1 Build Tests

C.2.2.1.1 Level 1

Build Acceptance Tests should take less than 2-3 hours to complete (15 minutes is typical). These test cases simply ensure that the application can be built and installed successfully. The objective is to determine if further testing is possible. If any Level 1 test case fails, the build is returned to developers un-tested.

C.2.2.1.2 Level 2

Smoke Tests should be automated and take less than 2-3 hours (20 minutes is typical). These tests cases verify the major functionality a high level.

The objective is to determine if further testing is possible. These test cases should emphasize breadth more than depth. All components should be touched, and every major feature should be tested briefly by the Smoke Test. If any Level 2 test case fails, the build is returned to developers un-tested.

C.2.2.1.3 Level 2a

Every bug that was “Open” during the previous build, but marked as “Fixed, Needs Re-Testing” for the current build under test, will need to be regressed, or re-tested. Once the smoke test is completed, all resolved bugs need to be regressed. It should take between 5 minutes to 1 hour to regress most bugs.

C.2.2 Milestone Tests

C.2.2.1 Phase I

The Phase I test plan tests C.1.4.1 and C.1.4.2 test cases to ensure that users can successfully create account, login to the application, and search for beers.

C.2.2.2 Phase 2

The Phase II test plan tests C.1.4.3, C.1.4.4, and C.1.4.5 test cases to ensure that users can successfully rate a beer, obtain recommendations on beers based on previous ratings, and share beer ratings with friends/fellow app users.

C.2.2.3 Phase 3

The Phase III test plan includes all previous tests to ensure all identified bugs have been remediated and that the application meet user acceptance criteria.

C.2.3 Bug Triage

The goal of triage is to define the type of resolution for each bug and to prioritize bugs and determine a schedule for all “To Be Fixed Bugs’.

Severity	Description
S1-CRITICAL	The defect affects critical functionality or critical data. It does not have a workaround. Example: Complete failure of a feature.

S2-MAJOR	The defect affects major functionality or major data. It has a workaround but is not obvious and is difficult.
S3-MINOR	The defect affects minor functionality or non-critical data. It has an easy workaround.
S4- TRIVIAL	The defect does not affect functionality or data. It does not even need a workaround. Example: Layout discrepancy, spelling error.

C.2.4 Test Completeness

100% test coverage

All Manual & Automated Test cases executed

All open bugs are fixed or will be fixed in the next release

C.3 Test Deliverables

Here are the sample deliverables,

a) Test Plan

g) Test Cases

h) Bug Reports

i) Test Strategy

j) Test Metrics

k) Customer Sign Off

C.4 Resource and Environment Needs

C.4.1 Testing Tools

Make a list of Tools like

a) Requirements Tracking Tool [Optional]

l) Bug Tracking Tool

m) Automation Tools

Required to test the project

C.4.2 Testing Environment

It mentions the minimum **hardware** requirements that will be used to test the Application.

Following **software's** are required in addition to client-specific software.

a) Windows 8 and above

n) Office 2013 and above

o) MS Exchange, etc.

2.

D. User Guide

D.1 Introduction

The purpose of this user guide is to demonstrate to users how to use and access the features of BeerTap.

D.2 Website Navigation

The welcome page (fig.1) is the first page users are greeted with when navigating to Beertap's website at https://beertapcmsc495.herokuapp.com/ (address will change in future)

fig1.

D.2.1 Searching for Beers and Breweries

Upon navigating to Beertap, users can immediately search for beers or breweries by selecting the corresponding navigation bar items. Upon clicking the user will be redirected to the appropriate search page.

For beer searches a beer's name in the search bar as shown in fig2(Left). where the beer "fat tire" is entered.

D.2.1.1 Beer Search Results

After entering a beer name to search, pressing the find beers button will return results from the search as shown in fig2(Right).

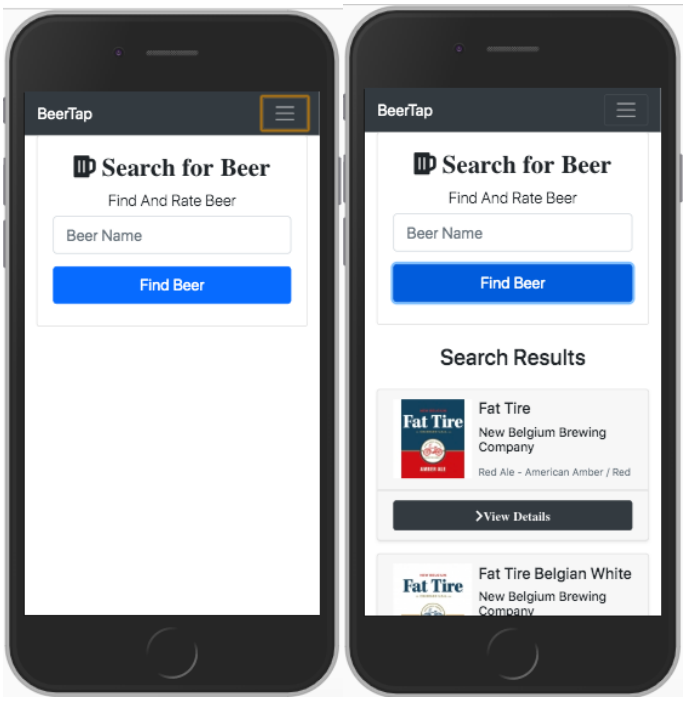

fig2.

D.2.1.2 Viewing Beer or Brewery Details

After Searching for a beer or brewery as shown in section D.2.2, users can view details by clicking the details button for the beer they wish to view as shown in fig3.

Clicking the button loads the details page for the selected beer. as shown in fig4.

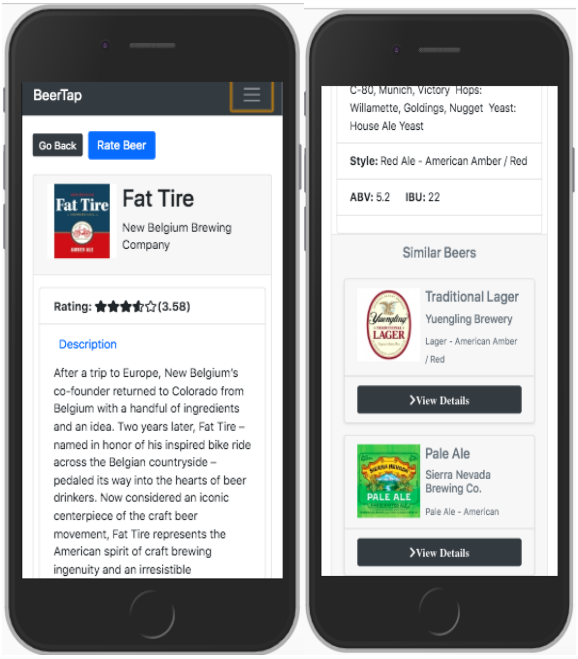

fig4.

D.2.2 User Login

To experience the full functionality of Beertap users must be logged in. Users can login from the navigation bar by clicking the login button in the top right as seen in fig1.

Upon clicking the login button users will see a be redirected to a login page where they can login with Untapped, through a facebook account or by signing up with untapped as seen in fig5.

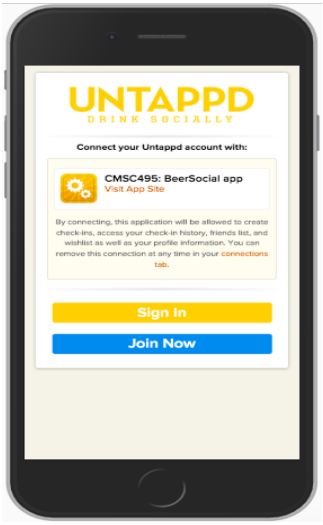

fig5

After completing the login users will be returned to Beertap's website. However, now the user will see his or her name on the welcome page.

By logging in the user will now be able to rate beers and share those ratings with friends.

D.2.3 Rating Beer

After navigating to the beers detail page as shown in section D.2.3 users can rate the selected beer by clicking the rate button as shown in fig4. Clicking the button will show a pop-up for the selected beer's rating as shown below in fig6.

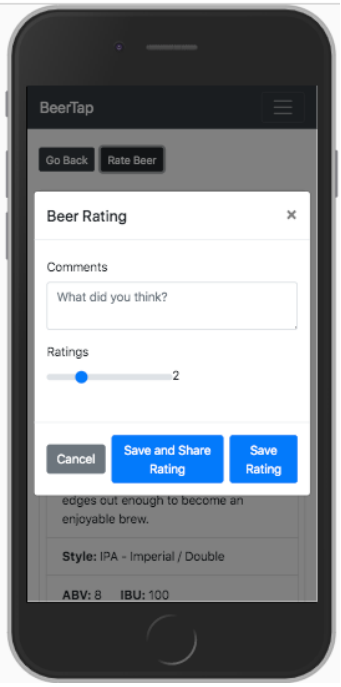

fig6

Users can then enter a comment about the beer or enter a rating*. Users can then view all their previous ratings by clicking the UserRatings option in the navbar.*

D.2.4 Sharing Beer ratings

Once completing the form, the user can either click the save rating button to save the rating and return to the details page or click the cancel button. If the share rating button is selected the user will see a Facebook popup dialog appear as seen in fig7.

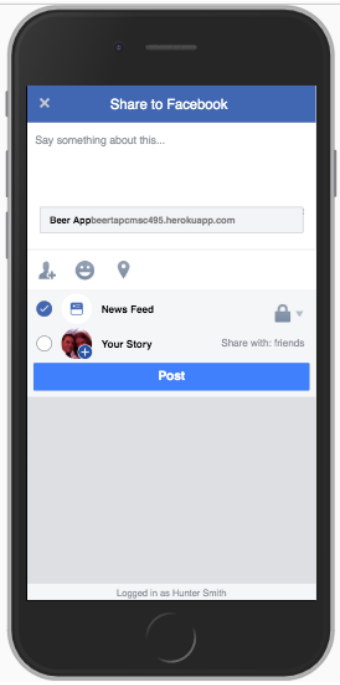
Share to Facebook
Say something about this...
Beer Appbeertapcmsc495.herokuapp.com
News Feed
Your Story
Share with: friends
Post
Logged in as Hunter Smith

fig7.

E. Phase I Report

E.1 Phase Milestones

Phase	Milestones	Status
1	Initial development of front-end, user interface created and deployed to test environment.	Complete
1	Functionality to allow user to create an account and login to the application	Complete
1	Functionality to allow user to search for beers	Complete

E.1.1 Frontend

The frontend, user interface, has been developed and deployed by frontend developer Hunter Smith. At current, the interface is stand-alone (communicating directly w/ the Untappd api); implementing the following requirements,

- Beer and Brewery search design functionality/requirement.
- User Login functionality/requirement.
- Similar Beer functionality/requirement. Currently not based on ratings input but beer type).
- Beer rating functionality/requirement. Currently not all beer attributes are implemented (Aroma, Taste, etc), only an overall rating with comments. Additionally,

the sharing feature is not connected to Facebook yet, but does share rating through the Untappd social network at https://untappd.com/)

The current implementation does not meet all requirements yet, as many of the requirements will only be implementable through a backend/database. During the next phase the frontend will switch from communicating directly to the Untappd api to the backend and further continue to fulfill project requirements.

E.1.2 Backend

placeholder text

E.1.3 Testing

Bug ID	Description
S3-01	Login button appears as a “hamburger” button when the screen is small. May be confusing to some users looking for the Login button.
S1-01	When clicked, the logout button does not log the user out or kill the session.
S2-01	When re-launching the application after clicking on the logout button and closing the browser, I am automatically logged back in when I click on the Login button. This bug will be fixed once S1-01 is fixed.

E.1.4 Summary

Project is on-track and progressing well. All Phase 1 milestones were met. Backend development was kicked-off with a video-conference call between Hunter, Steven, and Kevin (audio only) in which the path forward was discussed. The following adjustments to the project were made:

- Users will rate beers using a single attribute based on a number between 0 and 5 rather than rating multiple attributes such as taste, etc.

- Users will not be able to request recommended beers based on current location as the database will not have enough data to adequately test the functionality.

F. Phase II Report

F.1 Phase Milestones

Phase	Milestones	Status
1	Initial development of front-end, user interface created and deployed to test environment.	Complete
1	Functionality to allow user to create an account and login to the application	Complete
1	Functionality to allow user to search for beers	Complete
2	Validation testing to ensure bugs reported during Phase 1 testing have been mitigated	Complete
2	Functionality to rank beers and store data in backend database	Complete
2	Functionality to share beer ratings with other app users	Complete
3	Validation testing to ensure bugs reported during Phase 2 have been mitigated.	Incomplete
3	User Acceptance Testing	Incomplete

F.1.1 Frontend

Bugs Identified at the end of phase I have been corrected. Additionally, the ability for users to view all their rated beers from a single page has been added.

Bugs Identified at the end of phase I have been corrected. Additionally, the ability for users to view all their rated beers from a single page has been added.

Much of the rest of frontend development has been geared to ensuring a seamless transition from using untaped's api directly to using an intermediate backend. As such, the ability to share beer rating information is currently limited to the user sharing the information from there untapped account directly, and not supported by beer tap yet. This ability to share ratings within beerTap will be enabled upon backend/server deployment; Which has been rescheduled to next week.

F.1.2 Backend

Backend and MySql development and deployments were achieved during the week. While the backend service has limited RESTful functionality and serviced routes, it is now responsive to REST requests and makes the appropriate CRUD operations to the MySql Instance.

Additional work included a production server standup, additional Database deployment to the server, backend application deployment, endpoint binding, networking configurations, re-registration of public domain-name (etienne-monkey.com/).

Tasks	**Description**	**Status**
Schema Validation	Validating table design, primary/foreign key relationships and data to persist.	90% (additional schema changes occurring to match frontend requirements)

Database Standup	Database installation, configuration, and deployment on production server	100% (database is deployed and functional with current schema)
Service Object design	Strongly typed object design in service based on Database Schema	90% (additional changes to strongly typed objects as changes in design and schema occur)
Service Framework Development	Basic functional requirements for allowing the backend application to run, and separation of concerns following ioC principles	98% (architecture for running as a Windows application, accepting/ responding to HTTP requests, and database operations, all concurrently. Additional work related to proper shutdown of service desired as well as registering with the Operating System as a service.)
Service to DB communicatio n	Library installation and configurations for Dapper ORM, Database Connection string configuration for service, and dll import for handling connections to MySql instance through .NET 4.6.1.	100%
Service HTTP support	HTTP request, response, and error handling	65% (Additional work is needed to address routing and error handling to provide an informational response to the client but not expose the inner workings of the backend service)
Data persistence between Service and DB	Appropriate SQL queries and syntax used in Service to perform desired CRUD operations.	50% (Additional work required. Need to finalize remaining object repositories, object design, schema design, and use functional SQL queries instead of making numerous iterative DB calls)

Networking, Public domain	Network, Server, Router configurations and public domain name re-registration	85% (Configurations allow for server to host public domain name, and traffic is routed to server appropriately, however Networking/Server security needs to be increased without compromising functionality)
Service exposed	Service accepts and responds to HTTP requests to the domain that follow the supported URL routes	85% (Additional work related to error handling and unsupported calls desired)
3rd party API communicatio n	Service makes API calls to Untappd and BreweryDb to retrieve missing information, and update DB with User information, beers, and ratings.	0% (No work has been started on this)
3rd party API object design	Object design from 3rd party applications imported and translated to service's objects	25% (Untappd object design has been imported, BreweryDb has not, no adapter/translation work started.)

F.1.3 Testing

Bug ID	**Description**
S3-02	When rating a beer, the rating does not show up on the "User Ratings" page until after the user logs off and logs back in.

F.1.4 Summary

Project is ahead of schedule. All bugs identified during Phase 1 testing have been remediated and have been closed. All Phase 2 milestones were met. Only one minor bug was detected during testing. This puts the project ahead of schedule as less time will be required to be spent on remediating bugs and preparations for final testing can begin.

G. Phase III Report

G.1 Phase Milestones

Phase	Milestones	Status
3	Validation testing to ensure bugs reported during Phase 2 have been mitigated.	Complete
3	User Acceptance Testing	Complete

G.1.1 Frontend

Bugs Identified at the end of phase II have been corrected. Additionally, the ability for users to share all their rated beers on Facebook has been added.

G.1.2 Backend

Webservice/backend switched over to a VM. Endpoints support GET and POST rest requests and perform CRUD operations:

- /user/
- /reviews/
- /brewery/
- /beer/

Tasks	**Description**	**Status**
Schema Validation	Validating table design, primary/foreign key relationships and data to persist.	100%
Service Object design	Strongly typed object design in service based on Database Schema	100%
Service Framework Development	Basic functional requirements for allowing the backend application to run, and separation of concerns following ioC principles	100%
Service HTTP support	HTTP request, response, and error handling	100%
Data persistence between Service and DB	Appropriate SQL queries and syntax used in Service to perform desired CRUD operations.	100%
Networking, Public domain	Network, Server, Router configurations and public domain name re-registration	100%
Service exposed	Service accepts and responds to HTTP requests to the domain that follow the supported URL routes	100%
3rd party API communication	Service makes API calls to Untappd and BreweryDb to retrieve missing information, and update DB with User information, beers, and ratings.	100%
3rd party API object design	Object design from 3rd party applications imported and translated to service's objects	100%

G.1.2 Backend Modification

Although the initial backend design has been completed, a change was incorporated during phase III to utilize a smaller streamlined backend for the time being, leaving the more encompassing original backend for potential future improvements/implementations. The streamlined backend utilizes Node.js, Express Server, and MongoDB. The server is deployed on Heroku and provides CRUD operations for review data. The data design of this alternative is,

- checkin_Id: {type: Number, required: true, unique: true},
- checkin_comment: {type: String},
- first_name: {type: String, required: true, max: 100},
- last_name: {type: String, required: true, max: 100},
- bid: {type: Number, required: true},
- rating: {type: Number},
- created_at: {type: Date},

G.1.3 Testing

No bugs detected during testing.

G.1.4 Summary

All bugs identified during Phase II testing have been remediated and have been closed. All Phase III milestones were met. No bugs detected during final testing. Development and testing are complete.

H. Conclusions

H.1 Lessons Learned

Working as a virtual team presented some challenges that we were forced to address. For instance, when the project began, we were using the Team 3 discussion board for collaboration. We

noticed that this was not a very effective way to manage the project, so we started using Slack. Once we were able to chat in real-time with each other, the team became more productive. In addition, we used Google Drive to store documents which was much easier than trying to keep track of the current document in a discussion board. For discussions that required more interaction than chat messages, we used Google Hangout for "face-to-face" discussions where we could share our screens and discuss issues. Using these various collaboration tools was a game changer for our team and enabled us to complete the project on time.

Another lesson our team learned was to assign both a primary and alternate POC to each task. We had a team member stop responding and participating early in the project. This caused some issues with tasks not getting completed and the rest of the team having to fill in the void at the last minute.

We also learned that each team member had a different set of skills to offer the team but that the highly technical skills required for developing the app were only possessed by two team members. Through sharing the source code and each contributing to the project documentation, we were all able to understand the project and work together as a team.

H.2 Design Strengths

Our app was designed to be highly compatible with most mobile devices running a modern web browser. Modern web browsers support service workers that run separately from the main browser thread to cache content. This not only improves performance but also provides the capability for an "offline" mode where the user does not have an internet connection. This allows the user to refer to the list of beers that they have rated even if they are offline.

The app does not need to be installed by the user as it is available using the URL. This means that the user does not need to visit an app store to have the app installed. The same goes for updates. The app will automatically update when changes are published to the site. This capability ensures that all users of the app will be using the same version and multiple versions will not need to be supported.

H.3 Limitations

One limitation of the app is that in order to take advantage of the offline capabilities the user must be using a device with a modern web browser that supports service workers. If the user is using a legacy device/browser, the app will still work in the browser but the option to create a icon and cache the content will not be available to them. Another limitation of the app is that it relies on a third-party API for the beer details. Should that API not be updated or be discontinued, our app would be affected.

H.4 Future Improvements

Assuming the app was now going to enter the support phase in a production environment, future improvements would be based on customer feedback. One potential area for improvement would be to include multiple elements that the user could use for rating the beer. Another potential improvement could be the ability to create a social network within the app to share beer ratings with fellow beer lovers in the area.

H.4 Overall Contributions

Members	**Contributions**
Nishan Basnet	Contributed in creating the test plan, program testing, and documentation management
Kevin Hoy	
Steven E	
Hunter Smith	Created and coded the application's frontend and a secondary/backup backend, deployment of prototypes and final application, error/bug corrections, creation of design template, and general assistance in maintaining documentation.