10/31/2017

PROJECT PLAN

BREW DAY!

PREPARED BY: GROUP 7

Abhishek Nagaraj - 1211417906

Akshay Jain - 1211223907

Hari Iyer - 1211062148

Rundong Zhu - 1211537987

Sneha Vidhyashekar - 1211274841

Varuni Hullur Shama Rao - 1211222841

Table of Contents

Project Description 2
Overview (CO-5): 2
Key Requirements (CO-5): 2
Deliverables (CO-5, CO-6): 2
Acronyms and Abbreviations (CO-7): 2
Design and Architecture (CO-1, CO-3) 3
Implementation Strategy 3
High-level Work Breakdown Structure (CO-2): 3
Schedule / Timeline (CO-2): 3
Required Hardware (CO-2): 3
Third party content (CO-2): 3
Quality (CO-2): 3
Other Special considerations (CO-7, CO-3): 4
Process 4
Process Description and Justification (CO-2) 4
Tools (CO-2): 4
Roles and Responsibilities (CO-2): 4
Location of Project Artifacts (CO-2): 4
Sponsor Communications (CO-7): 4
Risk management 4
Identified Potential Risks (CO-2): 4
Mitigation Strategies (CO-2, CO-3): 5

Project Description

Overview (CO-5):

Home brewing, the process of producing beer on a small scale for personal purposes, is an activity that receives growing attention among beer enthusiasts. Every home brewer owns some brewing equipments like kettles, fermenters, pipes, etc. with a certain maximum brewing capacity, the number of liters the equipment can handle in a single "batch". Brewing also requires ingredients, whose actual amounts vary from recipe to recipe; which include various kinds of malts, hops, yeasts and sugars (and of course, water). Brewers like to log their recipes, for future reference, and maintain an updated list of available ingredients, for shopping before the next brew.

Brew Day! is an application that allows home brewers to maintain an organized database of their beer recipes. The application allows users to create, store and modify recipes, and later on delete them, if the user wishes to do so. The project implements features described above, i.e., creation, modification and deletion of recipes, creation of recipe instances (brews), support for notes on brews, and keeping track of available ingredients. The "What should I Brew Today?" is a mandatory feature. Ingredients availability feature can be implemented to be either manually entered by the user, or automatically compute from the brews information.

This project is a web-based application which uses HTML, CSS and Bootstrap for front end development; Java, Rest API and Apache Spark as a part of back end development. The application is developed with the focus on home brewers as the prime user.

This document gives a preliminary plan for how the team aims to achieve the above stated tasks. The first section gives an overview of the project along with the key requirements, describes project deliverables, lists the acronyms with their meanings that may be encountered in the rest of document. In the second section, design and architecture of the application to be developed is described. Third section includes the implementation details and strategies, with the details of high-level work breakdown structure, proposed timeline, required hardware, third-party content (if any), quality and other considerations. Fourth section entails the process description, tools used in the process, roles and responsibilities of each team member, and sponsor communications details. Fifth and the final section involves a well-thought specification of the identified potential risks, and mitigation strategies to overcome the potential risks.

Key Requirements (CO-5):

With the web application, the users will be able to maintain a database of brewing recipes, and the ingredients required for the brewing process. The following list offers a brief outline of the key requirements and major functions the application must perform or must let the user perform.

- A user must be able to view the recipes, i.e., providing user access to the recipe database
- User must be allowed to edit the brewing recipes
- User should be able to access the ingredients involved, and equipment needed

- User should have the ability to add new recipes to the database through the application
- Brewers should have the ability to make updations to the batch size
- Application should provide the option to delete the recipe as well
- "What Should I Brew Today?" suggests a brewing recipe for the day
- Update the ingredients, either automatically or manually by the user
- Store notes pertaining to each brewing recipe
- Store the brew log, when was that beer brewed; records of the quantity and time
- Application should be able to keep track of available ingredients
- Interface should be user-friendly, requiring no technical expertise from the users
- Project must be developed on open source technologies and platform
- Project must be offered under an open source license accessible through github or similar platform
- Project should be well-documented with documents including Software requirements specification, Software Design document, and a User manual

DELIVERABLES (CO-5, CO-6):

Following is the decomposition of the project into specific deliverables.

- Software Requirements Specification
- UI Prototype of the application
- Software Design Document
- Source Code
- System (application)
- System Testing/Integration Testing/Test scripts
- User Documentation(user manual)

ACRONYMS AND ABBREVIATIONS (CO-7):

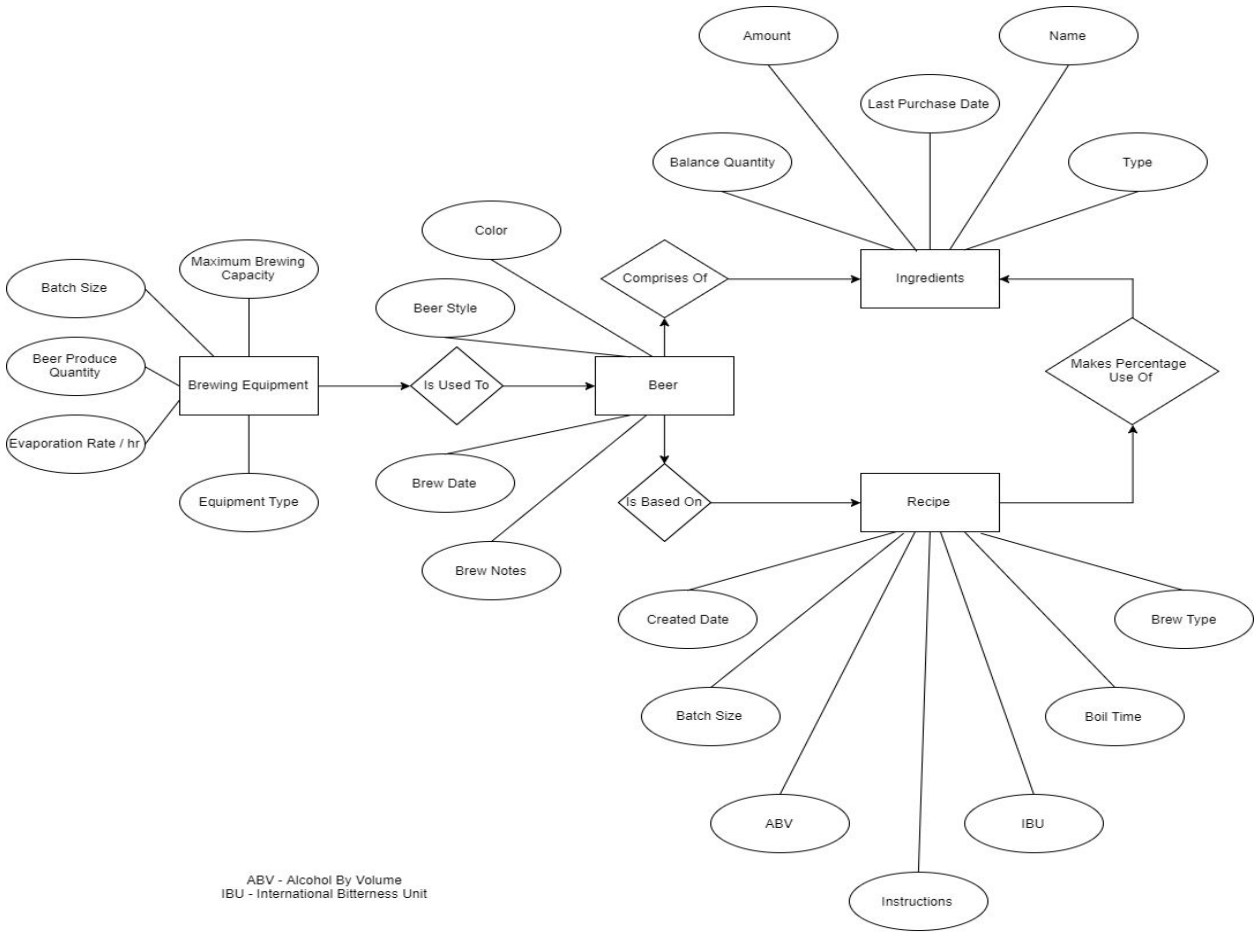
Term	***Definition***
HTML	Hypertext Markup Language (*HTML*) is the standard markup language for creating web pages and web applications.

CSS	Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language.
Bootstrap	Bootstrap is a free and open-source front-end web framework for designing websites and web applications.
Rest API	Representational state transfer (REST) is a way of providing interoperability between computer systems on the Internet. Application Programming Interface is a set of subroutine definitions, protocols, and tools for building application software.
UI	User Interface is a way through which a *user* interacts with an application or a website.
MVC	Model–View–Controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user.
E-R Diagram	An Entity Relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.
UML	Unified Modeling Language (UML) is a standard visual modeling language used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.
ABV	Alcohol By Volume(ABV) is a standard measure of how much alcohol is contained in a given volume of an alcoholic beverage
IBU	International Bitterness Unit (IBU) is a scale used to approximately quantify the bitterness of beer.

DESIGN AND ARCHITECTURE (CO-1, CO-3)

Brew Day project is a web application based on Model-View-Controller architecture. Overall design of Brew Day project focuses on helping users to make beer by themselves. For instance, user selects one type of beer, provide that as an input, and based on the selection, the system comes up with the corresponding recipe, ingredients involved and equipments needed for the brewing process.

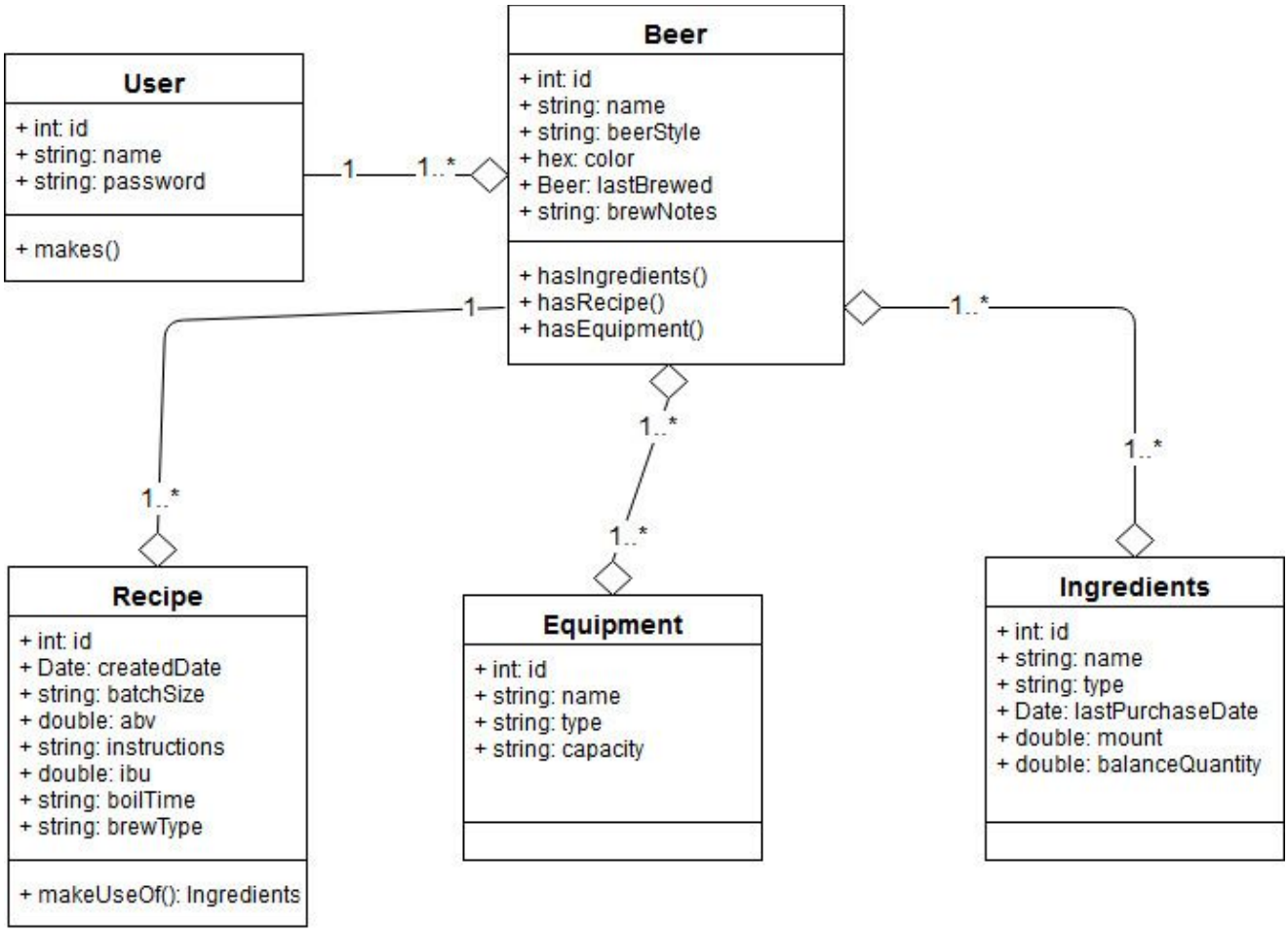
Relational Database Design: *E-R Diagram*



E-R diagram indicates that the main entities (rectangles) of the database are User, Beer, Recipe, Equipment and Ingredients. Linked with the corresponding actions (diamonds) among them, each entity has a set of attributes (ovals) uniquely associated with themselves. For example, entity 'Recipe' has attributes like Batch Size, Boil Time, ABV, IBU, and others.

Object-oriented Design: *UML Diagram*

- User: Design to identify each user in system which requires id, name and password. The function of user is makes Beer.
- Beer: Design to identify each beer user made which requires id, name, beerStyle, color, lastBrewed and brewNotes. The functions of Beer are hasRecipe, hasIngredient and hasEquipment
- Recipe: Design to identify each recipe for each beer made by user which requires id, createDate, batchSize, abv, instructions, ibu, boilTime and brewType. The function of Recipe is makeUseOf ingredients
- Equipment: Design to identify equipments used of each beer made by user, which required id, name, type and maxCapacity
- Ingredients: Design to identify ingredients used for each beer made by user, which required id, name, type, lastPurchaseDate, mount and balanceQuantity



Design Pattern:

- Pattern Name: Recipe Output
- Description: After user selected beer will brew, system must provide corresponding recipe and allow user to change it. Ingredients also change according to recipe, such as batch size, ABV, IBU and so on. Moreover, recipe result stores in database after creation, modification and deletion.
- Proposed Solution: Using Model-View-Control architecture, create recipe instance based on data model, modify recipe data on webpage and store final recipe in database.
- Consequence: Therefore, each instance of recipe models could be controlled by user and corresponding changes based on relationship implement by Hibernate. Finally, system shows the recipe as output according to user preference.

Implementation Strategy

High-level Work Breakdown Structure (CO-2):

- Software Requirements Specification (SRS) - Needs System analysis skills, 4 weeks
- UI Prototype of the application - Front-end development, 3 weeks
- Software Design Document - Software Design, 4 weeks
- Source Code - Programming and Software Architecture, 12 weeks
- System (application) - Interleaved with source code, would extend to a couple of weeks
- System Testing/ Integration testing / Test scripts - Software Testing, 5 weeks
  User Documentation (requested by the sponsor) - Needs User Experience analysis skill, 2 weeks

Schedule / Timeline (CO-2):

Overall milestones of the project can be summarised as follows:

1) Project Plan: Strategy and plan for the overall project progress which asks the brewers on what to brew on that particular day.
2) Requirements: Software requirements specification. Analysis and elicitation of requirements obtained from the sponsor and document the same.Update the changes or enhancements required as and when we progress with the project.

3) Website Front-end Design: Creating the prototype of the web-pages.Designing the Front-end part of the Website using HTML, CSS and bootstrap.
4) Database design: Designing the (E-R Diagram) database which contains all the recipes and ingredients of the beer and designing a database format.
5) Design documentation: Documentation of both the Database and the Front-end design
6) Backend implementation: Connecting front end to the back-end of the website using Java, Rest API and Apache Spark
7) Testing: Unit testing, integration testing - integrate web page, integrating front end with the back-end.System testing and document all the test results.
8) Final documentation: Which includes all details of the milestones. Creating the user manual once the system is up and running which can yield accurate instructions for use.
9) Maintenance: Implementing the changes as required by the user in the sprint cycles and maintaining the web page and the database.

Semester-wise breakdown:

Semester 1, Fall 2017: Project Plan, Requirements specification, Website Front-end Design, Database Design

Winter Break: Research about the Testing and Maintenance tools and technologies.

Semester 2, Spring 2018: Design documentation, Back-end implementation, Testing, Final documentation, Maintenance

Required Hardware (CO-2):

This is a web application designed for home brewers. The developers and the team members will have to have a simple computer with access to popular browsers like Google Chrome or Mozilla Firefox(for front-end development), and Java programming (for back-end development and testing). Users can access the application through the web, thus are required to have a computer or a tablet device, or a mobile device. This application does not require any specific programming language to be able to run on their systems, only requirement is the availability of the internet connection and a browser on the device, which is not any kind of extra effort to be put by the user.

THird party content (CO-2):

We do not plan to include any third party content in our project.

Quality (CO-2):

Primary quality goal of this project is to achieve all the milestones successfully by a timely release of all the deliverables mentioned above in the document. To achieve a high quality software, we will test, validate and verify each and every possible deliverable.

- We plan to meet the necessary requirements by verifying and validating them efficiently.
- We monitor the specific project result to determine if they comply with the system requirements.
- We will make sure that there are no defects or bugs in each of the deliverable through testing, including the final deliverable of the project. Hence, we achieve the built-in quality.
- We will make sure that we identify all the problems before the sponsor/user does.
- We will try our best to keep the user interface as user-friendly and easy to navigate as possible.
- We plan to rework on all the issues raised by the sponsor and meet the expectations.

To maintain the quality of the projects we use different tools like Control charts, Checklists and Flowcharts in our project. We use the customer satisfaction ratings (scale of 1 to 10) for each deliverable to the sponsor. Our team makes sure that the customer ratings are always high. We use the Product defect ratings (scale of 1 to 10) for each module of the project. This helps us keep track of issues / defects / bugs in each module and helps us rework on the modules that require rework.

Other Special considerations (CO-7, CO-3):

Our Project is a Web application. Additional considerations for the project maybe developing an android application for the same specifications.

PROCESS

Process Description and justification (CO-2)

Agile methodologies are used. All project artifacts are hosted on a public repository with issue tracking facilities (e.g., GitHub). The project's functionalities is thoroughly tested and tests are systematized using suitable unit testing libraries. In order to track the project's activity and progress, the following rules are being followed:

- For each major set of development tasks, a milestone characterising them is defined.
- For each development task, an issue describing them are defined, within the corresponding milestone.
- Task are broken down and distributed among the team members.
- Prior to creating a new issue, a checking routine is carried out throughout to verify if the same issue has been previously created. This is done to avoid work redundancy.

- The use of branches for working on partial solutions or larger issues is performed.
- All team members make sure their partial solutions do not break the system (tests that passed before must still pass, project must compile, etc.).

The above development process was discussed with the Sponsor and is approved by the sponsor.

Tools (CO-2):

Category	***Tools***
Web Development tools:	Java, Bootstrap, HTML, CSS, JavaScript
Database technology:	Apache Spark
Code Editors:	Visual Studio Code, WebStorm
Communication tools:	Slack, Skype
Version management tools:	Github

Roles and Responsibilities (CO-2):

All team members are involved in each development phase. Most of the work has been equally distributed among all the team members. Every member has worked on every aspect of the project. Roles among the team members rotate through the course of the project based on the current phase of the project. But we have individual team members proficient in specific tasks who will guide the team to achieve the tasks as mentioned below. Major roles played by each team member are:

1. Abhishek Nagaraj: Project Design, Project/Sprint Meeting Management, Part of backend development, front end development and documentation.
2. Akshay Jain: Requirement Engineer, Application design, Full-stack development, Integration, Documentation, Testing.
3. Hari Iyer: Database Manager and Primary Sponsor Contact Point.
4. Rundong Zhu: Back end developer
5. Sneha Vidhyashekar : Project Documentation, Project/Sprint Meeting Management, Documenting Minutes of Meeting, Requirements and conclusion of the meeting.
6. Varuni Shama Rao: Front End Developer, Technical Knowledge Engineer, Software Engineer (Database).

Location of Project Artifacts (CO-2):

1. GitHub
2. Slack

3. Smart Sheet
4. Google Team Drive (members are added to the team drive)
5. Blackboard

Sponsor communications (CO-7):

Communication Platform : Skype, Slack
Sponsor meeting frequency : Bi-Weekly Meeting on Skype

Risk management

identified Potential risks (CO-2):

The following are the potential risks to the project's completion:

- **"What should I brew today?" consistency** - One of the project's functional requirements is the ability to optimally analyze balance ingredients to come up with the a recommended brew recipe. However, this would be relative to the percentage ingredients for recipe. Implementing something like the 0/1 knapsack might overrule the optimal solution. It is a strong impact, with a potential incidence rate of about 5 per every 10 instances assuming the ingredients are halfway consumed.
- **Recipe Sharing Perspectives** - There needs to be a feature to share the brewer's recipe with others. This, however is a fairly open ended task. It can open the doors for multiple paradigms of data sharing. It is a thin margin that needs to be maintained between the requirement to make sure the project adheres to the best possible means to accomplish the task. This is a decision-oriented risk, and will decide the recipe share methodology every time the feature is used.
- **Domain-Specific Knowledge** - Conducting meetings with the sponsor, we get a good idea about the brewery terminology and the math that goes into it. However, there might be an impending risk of the team not knowing some of those concepts that might be coming up at a later stage. This is
- **Software Versions** - Apache Spark versions keep updating their programming library which alters crucial functionality. Running everything in synchronization with UDF execution capabilities in rapid succession could be a challenge. However, once fixed, the system should be sturdy. If this issue crops up, it would be a one-off phenomenon.

mitigation strategies (CO-2, CO-3):

The following are the mitigation strategies for the identified risks:

- **"What should I brew today?" consistency** - There needs to be a further granular algorithmic solution that would identify the percentage of ingredients instead of working from a boolean

standpoint. This is accounted for in the timeline, so this mitigation technique should not cause that much of a disruption in the project development phase.

- **Recipe Sharing Perspectives** - The team will decide on the methodology to be adopted to share recipe data. It could be screenshot-based sharing, textual transmission of intra-brew data, or maybe something even better that any of the member can come up with.
- **Domain-Specific Knowledge** -This is a risk that the team thought might not be completely mitigatable. However, the sponsor has shared a few online resources for us to have a look at which would give us a better idea of how home brewing works. This would be a learning process throughout development, and definitely poses some amount risk despite an on-the-fly learning approach. The solution that the team discussed was to try and grasp suo-motu cognizance of the requirement at hand, which would play a vital role in alleviating the potential misunderstanding from the terminology standpoint. This might lead to devoting more time to the learning process during the sprint so that it can balance out at the testing and maintenance phase by making sure proper implementation without misinterpretation of domain-specific data.
- **Software Versions** - The team agreed to the fact that well begun is half done. Thus, there would be proper documentation from the very beginning about the technologies in use, which would ease out the process of tracking software versions and compatibility. This would be a part of the development part of the timeline, taking advantage of the fact that it is too trivial a task to hinder the planned project schedule.