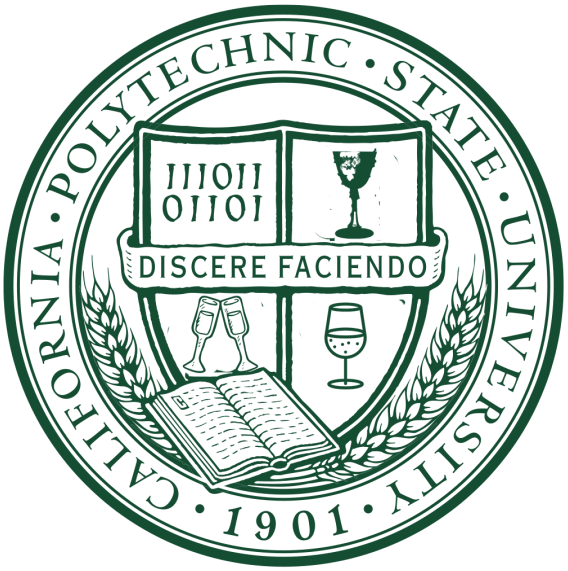
BLEND IT Applications and Server

Senior Project Report



Alex Bartlett, CPE
Aly Chapman, CPE
Tyler Fox, CPE
Cory Mayer, CPE
Rebecca McKinley, CPE

Client: Gregory Fields

Introduction

Project Overview

The BLEND IT Application Senior Project Team is a group of computer engineering students from California Polytechnic State University, San Luis Obispo. Our team was contracted by Gregory Fields, the owner of a startup wine blending venue called BLEND IT Winery & Bistro, to design and build software applications and a server that coordinate the ordering and dispensing of custom wine blends. We are also working in tandem with a Mechanical Engineering Design team who is building the Wine Blending Distribution System. The final product consists of one server and two phone applications: one application for the customer to create and order flights of wine blends and another application for the bartender to coordinate and fulfill orders. Through the customer application, customers are able to design unique wine blends for a tasting flight, name and save wines they like, and share their product with friends. Through the bartender application, bartenders are able to place custom blend orders, operate the Wine Blending Distribution System, and confirm transactions. We worked closely with our client as well as the Mechanical Engineering team to ensure both applications adhere to a high standard of code quality. Both applications are a product of extensive research and coordination between teams. With our final product, the team aims to have a positive impact on the community by creating an engaging new experience through a creative use of technology.

Client

Our client, Gregory Fields, requested our services to create applications for his company. While Greg's passion lies in winemaking, he possesses a thorough technical background. Greg has spent 10 years as an IT executive for a Fortune 500 company where he was responsible for over 24 computer data centers. He received his Bachelor's Degree in Computer Science and his Master's Degree in Information Resource Management. Through research and experience, Greg possesses an extensive knowledge of the wine industry. He has taken classes about viticulture and winemaking at both Cal Poly and Allan Hancock Community College and is working towards his Associate's Degree in Viticulture and Enology. Greg has done research into production and maintenance of commercially successful wineries. Through the BLEND IT Winery & Bistro, he wants to create a unique dining experience where customers have more control over the wine they taste. Our applications and server allow Greg's customers to create their own wine blends and share blends with friends. His customers are also able to buy bottles of wines they create through the blending process and receive discounts based on the popularity of their blends. Greg will receive and distribute the respective applications to his bartenders and customers who will then use them to create unique wine combinations.

Stakeholders

BLEND IT Winery & Bistro Customers

The customers of BLEND IT Winery & Bistro will primarily be millennials living in San Luis Obispo. While there are no BLEND IT Winery & Bistro customers at this time, the application must meet their eventual needs. The customers will only have access to the final version of the customer application, which they will use to order wine blends.

Mechanical Engineering Design Team

The Mechanical Engineering Design Team plans to create the machine that physically blends wines. The BLEND IT Application Senior Project Team will work closely with the Mechanical Engineering Design Team to ensure that communications between the respective systems succeed.

San Luis Obispo Restaurants and Nightlife Businesses

San Luis Obispo Restaurants and Nightlife Businesses include both potential partners and competitors. BLEND IT Winery & Bistro may affect these businesses, so these businesses are considered stakeholders. The deliverable to San Luis Obispo Restaurants and Nightlife Businesses is the whole BLEND IT Winery & Bistro system. The BLEND IT Application Senior Project Team provides mobile application software and software infrastructure.

BLEND IT Winery & Bistro Bartenders

Bartenders of BLEND IT Winery & Bistro have a stake in both the bartender application and the customer application. Meeting the customer requirements and making both applications as easy to use as possible will make the bartenders' jobs will be made easier. A well-made customer application means

more effective communication with customers for bartenders. A well-made bartender application produces more efficient bartenders.

BLEND IT Application Maintainers

BLEND IT application maintainers are people that will be maintaining and developing the BLEND IT applications after this Senior Project course is over. This group of people may be current BLEND IT Application Senior Project Team members, local computer programmers, telecommuting computer programmers, or outsourced computer programmers. Maintainers are essential for success because they will ensure that the applications continue to meet business requirements. Creating maintainable software will make maintainers' jobs easier. Maintainable software typically has tests, good code style, and is written such that modifications are not difficult. The maintainers' needs for maintainable software matches well with the BLEND IT Application Senior Project Team's desire to build maintainable software (as per the team's mission statement).

Wine Industry

The wine industry in San Luis Obispo County is very strong, and BLEND IT Winery & Bistro is a member and potential influencer. Wineries are both potential suppliers and competitors. Wineries could supply wine for the restaurant, but any wine sold in the restaurant could also detract from their market-share. Interest in wine blending created by BLEND IT Winery & Bistro might influence trends in the local wine industry. BLEND IT Winery & Bistro may lead companies to produce blended wine products because due to good performance or lead them away due to poor performance.

Framed Insights and Opportunities

The main source of both our engineering and customer requirements was our client, Greg. Throughout the Capstone and Senior Project classes we set up regular meetings with him where we demonstrated our progress on the applications and server. Through these meetings we formed a more coherent picture of what the system should look like and how to create feasible goals for the rest of the year.

One of the biggest shifts in our understanding of how the applications would function in the system came a few weeks into our planning. Initially, the customer would choose specific wines they wanted to blend and the wine amounts would be sent to the machines based on the customer's specifications. However, due to intellectual property concerns and the interest in creating a unique system, Greg decided that there would be a new blend selection process. This new process requires customers to choose from eight core wine flavors defined by experts: earthy, woody, caramel, nutty, herbaceous, fruity, spicy, and floral. This was a significant change that resulted in a new set of prototypes. However, it also came early enough in our process that it did not have as big of an impact on development as it could have if it came later.

In terms of the applications and the server, our understanding of the core functionality has not changed through our meetings with Greg. However, there has been a significant shift in the focus and look for the applications as a whole as development progressed. After our first prototype implementation, it was clear that there needed to be a better focus on flow and accessibility. This was not our focus initially, but after some meetings with Greg we had a better understanding of the user interface goals of the applications, especially the customer facing one. The application needed to appeal to all customers, including those who are not as familiar with newer technology. Minimizing the number of clicks to complete a core action was the next priority. Our meetings with Greg regularly offered checks on our progress and insight into our goals and development.

Project Goals and Objectives

As a Capstone group we developed these goals and objectives for the overall project as a metric for the standards and quality of the final product. We discussed several different possibilities for what we wanted to accomplish as the full Capstone group and as the smaller Senior Project group. Initially, we created very abstract goals in order to encompass everything that we thought the project would need. However, after receiving feedback and after the requirements of the project became clearer, we created more focused and specific goals. Objectives specify measurable ways to see how the goals progress. Our goals are listed below.

Goal:	Develop a single code base for Android and iOS platforms
Objectives:	-Choose an infrastructure and technology that utilizes code re-use -Have a single code base with no platform specific code -Implement a reliable framework that performs consistently on both platforms
Goal:	Ensure that project is easily scalable
Objectives:	-Have a sense of abstraction when coding - no hard coding values -Have enough redundancy for protection -Consider hardware limitations - do not ask for information it cannot provide
Goal:	Produce a product that meets the engineering and customer requirements
Objectives:	-Implement software testing as the product grows -Remain in contact with client and beta testers

-High code quality

-Have a code review process that is reliable and useful

-An intuitive UI/UX for the customer

Goal: Create an easily maintainable code base that does not require maintenance

Objectives: -Utilize a version control system for clarity and reliability

-Security considerations for customer data

-Have thorough documentation - anyone should be able to pick up the project

In Capstone we met several of these goals. We currently have functioning applications that run on both iOS and Android through a single code base and we are able to download the applications onto mobile devices. On the server, a viable code test pipeline has been implemented for software tests, as well as the ability for the project to be scaled as more machines or locations are added to the database.

In Senior Project, we met our client's goals for our applications and the server. We were able to implement the features he deemed important, such as the ability for a customer to place an order and to have that order make it through the bartender application to the Wine Blending Distribution System built by the Mechanical Engineering team. Any features we were unable to fully implement were stubbed out in the code for future developers to easily pick up. During the course of Senior Project we remained in contact with our client and testers to make sure the user interface was usable and self-explanatory.

Project Outcomes and Deliverables

At the end of our project we are handing over two working mobile applications, one for BLEND IT Winery & Bistro customers and the other for BLEND IT Winery & Bistro bartenders, as well as a server to connect these two applications and the Wine Blending Machine created by the Mechanical Engineering team.

Both mobile applications are available on Android and iOS systems and are connected to the server. They allow customers to place orders and allow bartenders to fulfill them. On the customer application, the collection of pages a customer goes through to place an order (what we call the Blend Wizard) is complete. A customer uses sliders on the first page of the Blend Wizard to select the proportion of flavors they want to taste. They then submit it to the server. Currently, the server returns data with the wines in stock and the customer application generates random blends to display. The customer then selects fives blends to create a flight and submits the order to the server before being redirected back to the main page. The news feed, popular blends, profile and login pages are stubbed out and statically displaying predefined data instead of pulling from the server.

On the bartender application, the core functionality is completely implemented. It involves the bartender viewing pending orders and sending them to the Wine Blending Machine. On the Pending Orders page, clicking on an order brings the user to an expanded page where a bartender can view the details of each blend including wine type and amount. If the bartender clicks the “Send to Machine” button, they are give the option of which machine to send the order to, a verification of the machine they chose, and a confirmation that the order was sent to the given machine. Bartenders are also able to check on the amount of wine left in each keg on the completed Wine Stock page.

The core functionality of the web server is implemented and working. The database is populated with descriptive and useful test data, and the web server services both mobile applications and the wine blending machine. As a customer steps through the Blend Wizard, information is either stored or

modified in the database in real time. Once an order is submitted on the customer application, it is then available for the bartender to see. After the bartender submits an order, it becomes available to the blending machine. Once the order has been fulfilled on the machine, the order is marked as complete and the system is ready for the next order.

The web server has around 15 API endpoints that the two mobile applications and the wine blending machine are using in order to make this happen. In the future, an algorithm will need to be developed that takes a customer's' selected flavor profile and returns possible blends. Also in the future, other small aspects of server functionality will need to be implemented (confirming a customer is physically inside BLEND IT Winery & Bistro, customer application submitting a table number with an order, etc.).

The web server code is all stored on a GitHub repository that automatically rebuilds, tests, and deploys our code every time something in the repository is checked in. Additionally, the web server can be run from the AWS console, which allows for easy use of AWS services using GUIs. Finally, the AWS command line interface can be installed on the future developers' computer and will allow them to work from the command line to leverage AWS services.

Background

The BLEND IT system is a complex project consisting of many different technologies and workflows. The customer application, the bartender application, and the Wine Blending Distribution System all communicate with a backend server which uses databases and some limited cloud computing.

The customer application allows customers to order custom wine blends based on flavor preference. The bartender application allows the bartender to view customer orders and send those orders to a wine blending machine for fulfillment. Both applications are written in a rather new language called Typescript and built with the React Native framework. Unlike native applications, there is no need for separate Android and iOS development teams, and unlike web applications, React Native accepts popular web components and generates their Android and iOS analogs as a true native application.

Amazon Web Services, also known as AWS, will handle the entire backend server [R1]. AWS allows our client the liberty to focus on his business without worrying about server management, security, or authentication. Greg has done network management for many years and utilizing AWS is critical for keeping the focus on providing customers with a unique experience instead of application maintenance.

BLEND IT Winery & Bistro has no direct competitors in terms of creating personalized blends of wine commercially, but there are other players in the viticulture space that offer a subset of the services that BLEND IT Winery & Bistro offers. Blendtique [R2] offers a web application to order a bottle of a custom blend of Merlot, Cabernet, Syrah, and Grenache. Blendtique also offers custom labels at no additional cost. The BLEND IT application will feature custom blending as well as custom labels. Vinfusion [R3] is an application-controlled system to make custom wine blends based off of a user's taste. The blending algorithm is based off of extensive customer research to match tastes to wines, as well as spectral analysis of chemicals in wine that give it a certain taste. With Vinfusion, four specific wines and three parameters are used to create a large variety of blends. The BLEND IT application will accept more parameters, and use more wines. Lastly, wine bars are very popular throughout the California

Central Coast area, though none offer an experience quite like BLEND IT Winery & Bistro. Though the BLEND IT applications' services are offered through its indirect competitors, the BLEND IT applications' solution will embody aspects of all of these products in ways others have not, and will bring a unique business into the Central Coast wine space.

Engineering Specifications

Engineering Specifications

The following engineering specifications for the customer and bartender applications were developed from a combination of customer requirements designed by the BLEND IT Capstone Team, our client, and our project goals.

Bartender Application

Spec. Number	Parameter Description	Requirement or Target with units	Tolerance	Risk	Compliance
1	Response Time	10 ms	Max	M	T, A
2	Flight Size	2.5 oz	Exact	L	T
3	Glass Size	9 oz	Exact	L	T
4	Bottle Size	750 mL	Exact	L	T
5	Time without Connection	10 minutes	Max	L	A
6	Code size	5000 lines	Max	L	A
7	Text localization	English	Exact	M	A
8	Software distribution license	Proprietary	Exact	L	A

Table 1: Bartender Application Engineering Spec table.

Customer Application

Spec. Number	Parameter Description	Requirement or Target with units	Tolerance	Risk	Compliance
1	Wines/blend	5 wines / 2 wines	Max / Min	L	T
2	Average time spent in Blend Wizard	2 minute	Max	M	T
3	Time to validate user's location/age	30 seconds	Max	M	T, A
4	Blends/flight	5 blends	Exact	L	T, A
5	Response Time	10 ms	Max	M	T, A
6	Time without Connection	10 minutes	Max	L	A
7	Steps to Create a flight	5 Steps	Max	L	A
8	Code size	7000 lines	Max	L	A
9	Text localization	English	Exact	M	A
10	Software distribution license	Proprietary	Exact	L	A

Table 2: Customer Application Engineering Spec table.

Use Cases

The following two use cases address the core functionality for the customer and the bartender applications. All other use cases can be found in Appendix [5].

Bartender Application

Use Case ID	3
Use Case Name	Bartender fulfills order
Actors	1. Blending machine 2. Web server 3. Bartender
Description	Bartender communicates with the blending machine to fulfill an order
Preconditions	1. Order is placed 2. Machine is active
Postconditions	Bartender fulfilled requested order
Normal Flow	1. Bartender selects order from bartender application 2. Order is sent to web server from bartender application 3. Web server sends the first pour to the machine 4. Bartender places glass on machine 5. Bartender confirms pour on machine screen 6. Machine dispenses pour and displays order being dispensed 7. Machine confirms pour completion with web server 8. Bartender removes glass 9. If there is another pour in the order web server repeats process starting at step 3

Customer Application

Use Case ID	1
Use Case Name	Order a flight
Actors	1. Customer 2. Web server
Description	Customer orders a flight of 5 blends
Preconditions	1. Customer has an active session 2. Wine chosen for blends is available
Postconditions	Bartender receives order
Normal Flow	1. Customer opens the initial blending process page from main page 2. Customer creates a tasting profile by moving sliders for how much of each flavor (nutty, caramel, fruity, floral, herbaceous, woody, earthy, and spicy) they want 3. Flavor profile created by the user is sent to the web server 4. Web server responds with 10 blends 5. Customer selects 5 blends to create a flight 6. Customer orders the flight 7. The order is submitted to the web server 8. Web server confirms that the order is submitted

Design Development

The final goal of this project is to have functioning applications for both the bartender and the customer and a server, which connects everything together. We have currently met this goal. There is a working product covering the main functionality of the bartender application, customer application, and server.

For the customer application, we started by only developing the Blend Wizard. At first it only involved the customer selecting two to five wines and creating five blends with them. This required the customer to go through a total of seven screens to place a single order. In order to simplify this process and avoid encroaching on existing patents, Greg decided the customer should instead choose the flavors they want in their blends. This streamlined the process down to three screens: one where the user first determines the flavors they want in their blends, one where they select five blends generated by the server, and one to submit the order. After completing the main functionality, we added the ability to go back to a previous page in the Blend Wizard and the ability to cancel an order.

After the Blend Wizard was created and partially connected to the server, we focused our attention on the next priority. We decided stubbing out the main page and login would be next. We currently have the main pages, including the news feed, profile, and login pages, pulling static data defined in the customer application to stub out how these pages should look when they are fully connected.

For the bartender application, there is a detailed process allowing the bartender to view pending orders and fulfill an order as well as full connectivity to the server with dynamic updates. Initially, the method of dynamic updating was not clear. It was possible to have the user exit the page and re-enter in order to trigger an update, or for the user to pull the screen down to update. However these methods required additional maintenance by the user. Our client wanted the Pending Orders page to be the one that

bartenders would leave open while working. In order to accommodate this use, we implemented a timer that would dynamically update the page with new information from the server.

Additional bartender application development changes came from working with the Mechanical Engineering Design team to specify how they wanted an order to count as fulfilled. This could happen on the bartender application, where a bartender clicks a “Complete Order” button, or on the interface with the Wine Blend Distribution System, or automatically from the machine once the channels are finished pouring. After some discussion, we decided that the order completion would be covered via the machine interface instead of the bartender application. It would be too difficult for the bartender to have to walk back and forth from the application to the machine when pouring a flight, and it would be impossible for the bartender to set up a glass at regular, timed intervals if fulfillment was managed automatically by the machine.

In terms of our tools, the user is able to run the application on any mobile device. Because the application does not require any iOS- or Android- specific technologies, it does not require access to any hardware that can only be exposed by native code. Both applications are cross-platform and written with React Native. React Native offers the ability to deploy a real native application to Google Play and the iOS App Store. This is in contrast to a web application, which only runs in a browser. React Native also allows for easy testing without recompilation. This is due to the native components being generated at runtime. React Native also has tools for iOS and Android that allow developers to test code as a physical app running on a virtual or a real device.

The selection of the database is key in how the business will function in years to come. A NoSQL database was selected for its flexibility. Not every entry needs to contain the same fields as all of its neighbors in a NoSQL database. NoSQL also allows for the immense growth of the database without needing to worry about segmentation. It also allows any field to be queried. At first, MongoDB was considered because it is extremely popular as well as open source. Our client will not want to deal with

running the actual server on a physical machine, so various cloud services, such as Firebase and Azure, were examined. After careful deliberation, DynamoDB, a tool from AWS, was selected. AWS offers a wide variety of useful services. DynamoDB is one such service that integrates with all of the AWS services including Cognito and Lambda. DynamoDB will store data pertaining to all customers, transactions, wines, blends, establishments, machines, and more. Cognito handles user identities and will only be used for setup and login purposes. Customers will be able to set up an account and log in with Cognito handling the credential provider details. Using the Facebook SDK and the Google+ SDK, users will be able to log in and set up an account. This generates an entry in a Cognito Federated Identity Pool. Using the social media SDKs, profile data is exposed and can be entered in the federated identities.

Lambda is another service provided by AWS to run code in the cloud. Lambda offers the ability to write NodeJS, Java, or Python code that runs in the cloud. A Lambda function, for instance, will handle user account setup. When a new id appears in Cognito, the lambda function will get more user data and securely store it in the DynamoDB.

BLEND IT is a large project that utilizes many technologies. It is unclear what the application experience will become, what extra data or functionality will be required, or the nature of any future roadblocks. Thus, it is possible that more services and technologies than those already listed will be included in the future.

Final Detailed Design

The final high-level design is diagrammed below in Figure 1. The BLEND IT Senior Project Application Team continued building the customer application, the bartender application, and the server after the completion of Capstone, while the Mechanical Engineering team built the Wine Blending Machine.

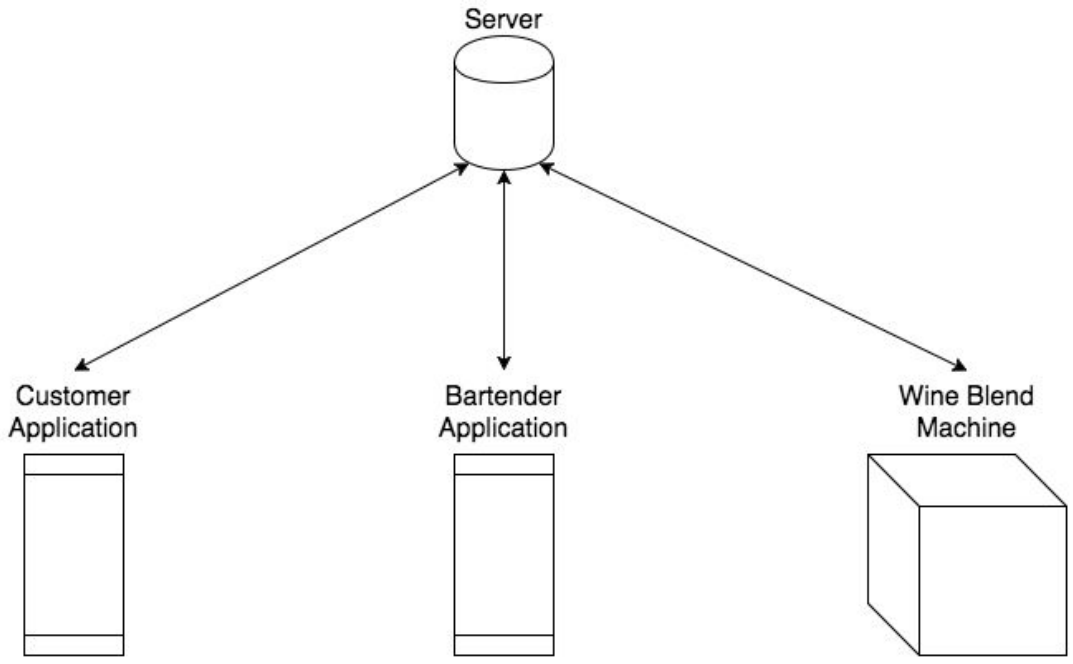


Figure 1: High-level design of software/hardware architecture

The customer application interfaces with the customer either through tablets in the BLEND IT location, or through the customer's personal phone. Customers enter their preferred flavors into the customer application, which then sends all information to be stored in the server. The server stores everything related to a customer's profile and chosen wine order. Employees of BLEND IT Winery & Bistro utilize the bartender application behind the bar. The application pulls information related to pending orders and displays them for the bartender's use. When a bartender is ready to assemble the blend, they will indicate through the bartender application, which sends a trigger to the server. From

there, the Wine Blending Machine pulls the order specifications from the server and dispenses the wine blend.

The above high-level diagram in Figure 1 was given to us as part of our client's preferred structure. We were not heavily involved in the design of the high level system. However, in order to accommodate the structure, the server team designed the schema of the database.

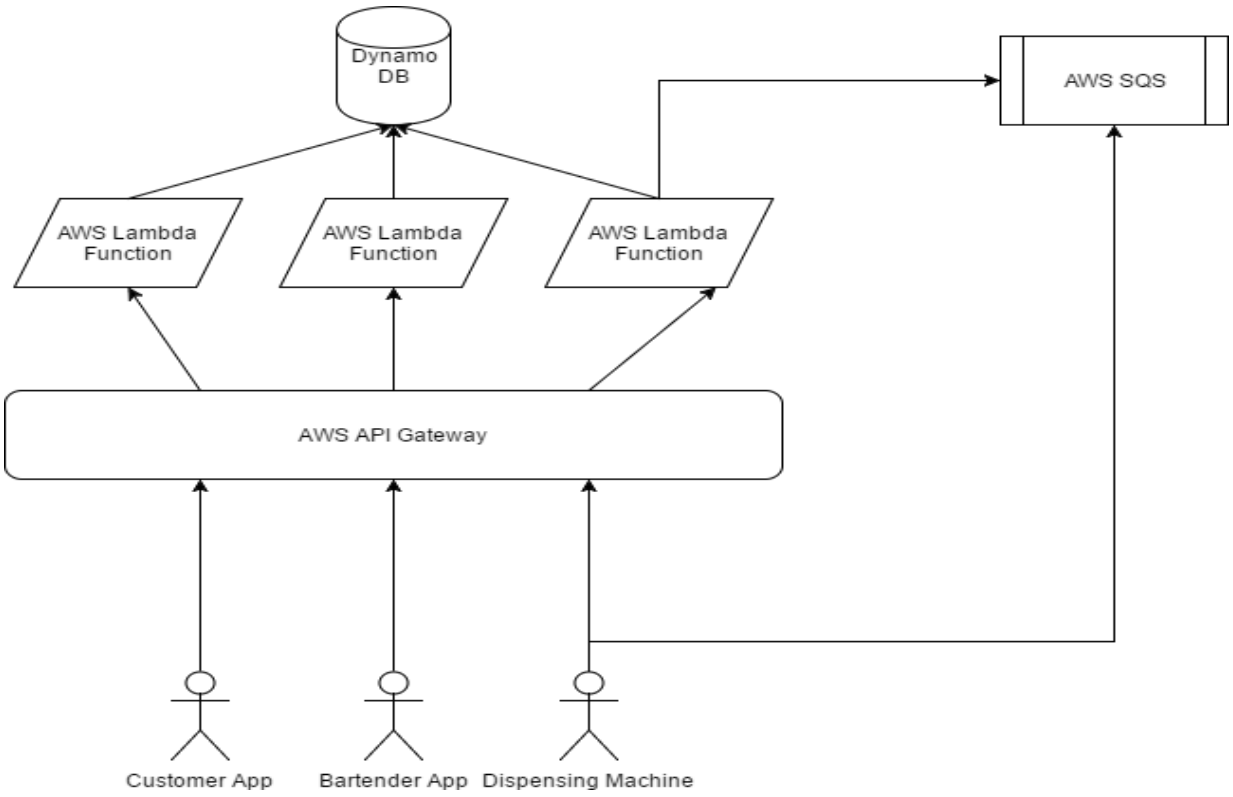


Figure 2: High level server architecture

Figure 2 above is a diagram of the server architecture. We use Amazon API Gateway to broker HTTP requests from consumers into our lambda functions. The API Gateway is able to authenticate requests to ensure the caller is valid before passing execution to AWS Lambda. It also provides a centralized URL for the different clients to communicate with the server. All of the lambda functions are able to access the database in order to return data to clients. Some lambda functions will also place messages in the Simple Queue Service. These messages will instruct the dispensing machine to pour. The

dispensing machine will directly poll the queue for new requests to fulfill them in the order they were received. The dispensing machine will also communicate to the server after dispensing in order to determine the failure or success of each pour.

The structure was designed to be easily scalable and to incorporate a modular design. This ensures that if an aspect of the design needs to be changed or updated, it will have as small of an impact as possible. Some of the tables contain a foreign key relationship with other tables. For example, the Session table contains a field called “customerId” which is a foreign key pointing to an entry in the Customer table called “userId.” Table 3 below contains the schema for the most important tables as well as some future tables for scalability. Though this is a schema-less design, these example fields are provided to give the system administrators a better understanding of what information is being passed between the tables and how it fits together.

Customers

Hash Key: CustomerId

Sample Item:

```
{
       CustomerId: “UUID”
       Email: STRING,
       FirstName: STRING,
       LastName: STRING,
       DateOfBirth: “String”

}
```

Wines

Hash Key: WineId

Sample Item:

```
{
       WineId: “UUID”,
       Name: STRING,
       Maker: STRING,
       Year: STRING,
       Type: STRING,
       Description: STRING,
       FlavorProfile:{
          Nutty: 50
          Earthy: 20
          Floral: 70
          Herbacious: 12
          . . .

       }

}
```

Blends

Hash Key: BlendId

Sample Item:

```
{
   BlendId: "UUID",
   CreatorId: "UUID",
   BlendName: "STRING",
   Rating: "NUM",
   Description: "STRING",
   LabelImageUrl: "STRING",
   Wines: [
      {WineId:"UUID",
       Percent:INT},
      {wineId:"UUID",
       Percent:INT}
       ...]
   }
```

Orders

Hash Key: OrderPlacedDateTime

Sample Item:

```
{
   OrderPlacedDateTime: "DATETIME",
   OrderId: "UUID",
   CustomerId: "UUID",
   OrderType: "STRING",
   IsFulfilled: "BOOL",
   Pours: [
      {
         BlendId: "STRING"
         AmountML: "NUM",
          Wines: [
             {
                 WineId: "STRING",
                 Percent: "NUM"
             },
             {
                 WineId: "STRING",
                 Percent: "NUM"
             },
           …
          ],
       },
       ...
   ]
   MachineInfo: {
      MachineId: "UUID",
      MachineName: "STRING",
      IsOnline: "BOOL",
      Location: "STRING",
      WinesOnTap: [
         {
            WineId: "STRING",
            IsOnline: "BOOL",
            RemAmountML: "NUM",
            TapNum: "NUM"
         },
         ...
      ]
}
```

Machines	Establishments (FUTURE)
Hash Key: MachineId Sample Item: ----------	Hash Key: EstablishmentId Sample Item: ----------

```
Machine1 =
{
   MachineId: “UUID”,
   IsOnline: “BOOL”,
   Location: “STRING”,
   MachineName: “STRING”,
   WinesOnTap: [
      {
         WineId: “STRING”,
         IsOnline: “BOOL”,
         RemAmountML: “NUM”,
         TapNum: “NUM”
      }, ...
   ]
}
```

```
{
      establishmentId: “UUID”,
      machines: [“UUID”, “UUID”]
      seats: [
{seatId: “UUID”, name: STRING},
{seatId: “UUID”, name: STRING}]
}
```

Table 3: Database schema tables for server

Signup and Login

Signup requires email, first name, last name, phone number, birthdate, username, and a password. Upon successful sign-up, a confirmation code is sent via email. The confirmation code will also serve to verify a valid email.

As of now, login is not fully implemented. While some development was finalized for the login architecture during the Capstone class, it was not properly communicated to the Senior Project team. Additionally, our client's main goal for the final quarter was to fully integrate the Blend Wizard across all components. For this reason, login was not a priority. As of now, we have login working on an isolated device using React Native, but it is not fully integrated into the customer application. For the demonstration, we will treat the customer application as if a user is already logged in.

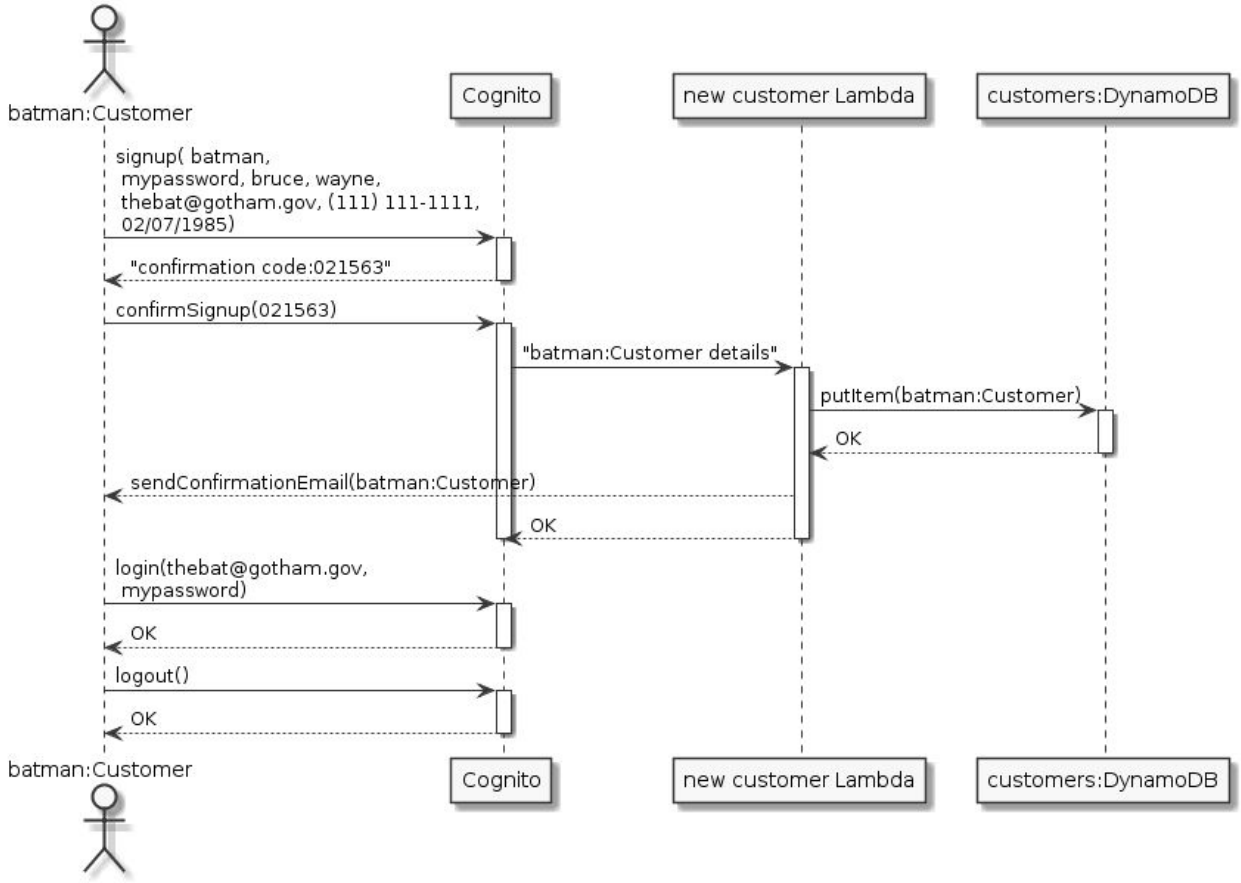


Figure 3: Login architecture for customer

Customer Application Architecture

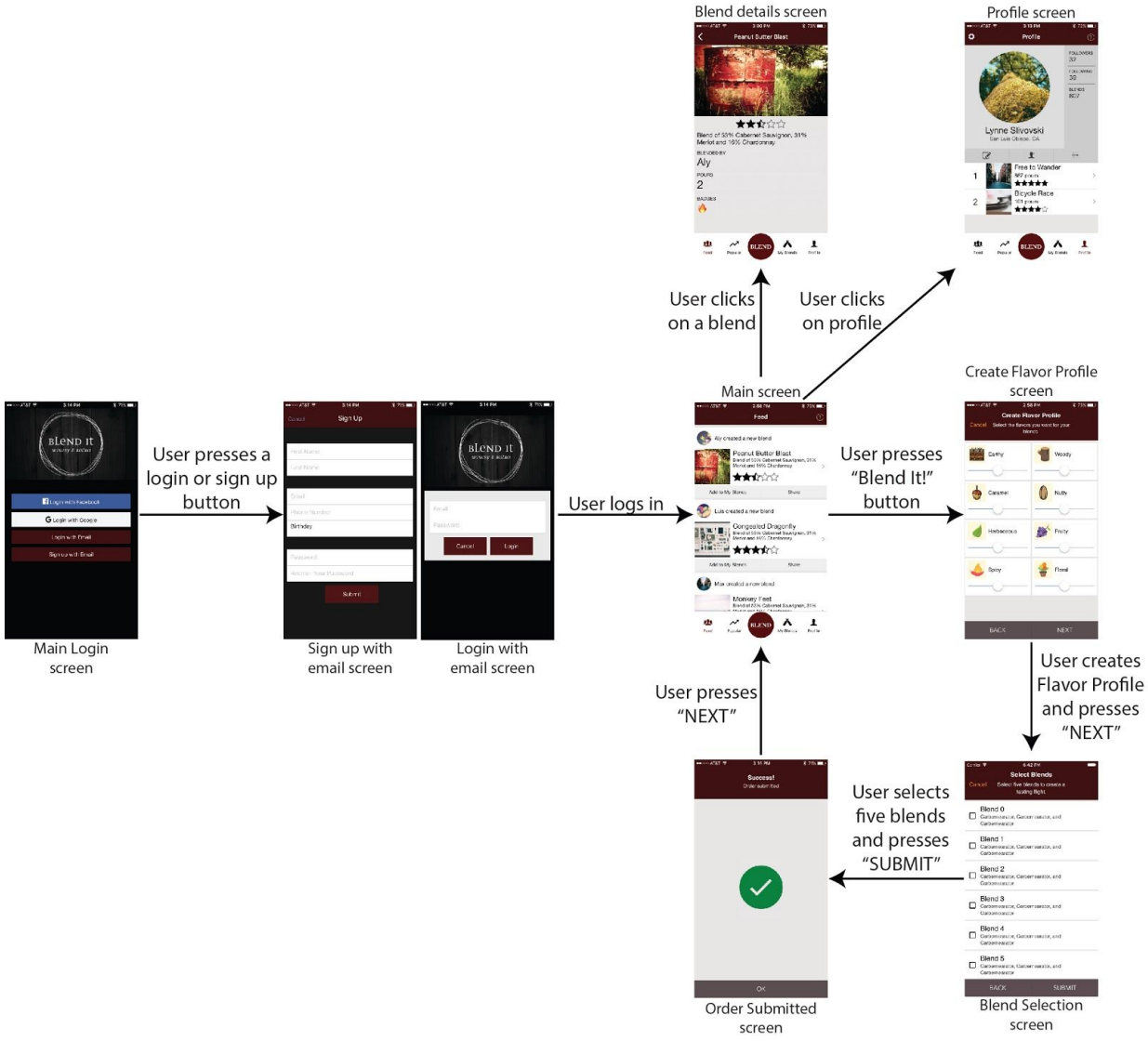


Figure 4: Basic customer application flow

The above diagram details the main pages in the BLEND IT customer application that have been designed. The basic flow of the applications begins with the customer at the login screen choosing a method of login. After login, the customer is directed the main screen where the "feed" page is displayed. From there, the user can click the "BLEND" button and order a flight of wine through the Blend Wizard, or they can choose to view details on blends listed, see popular blends, view their profile, or see saved blends.

Bartender Application Architecture

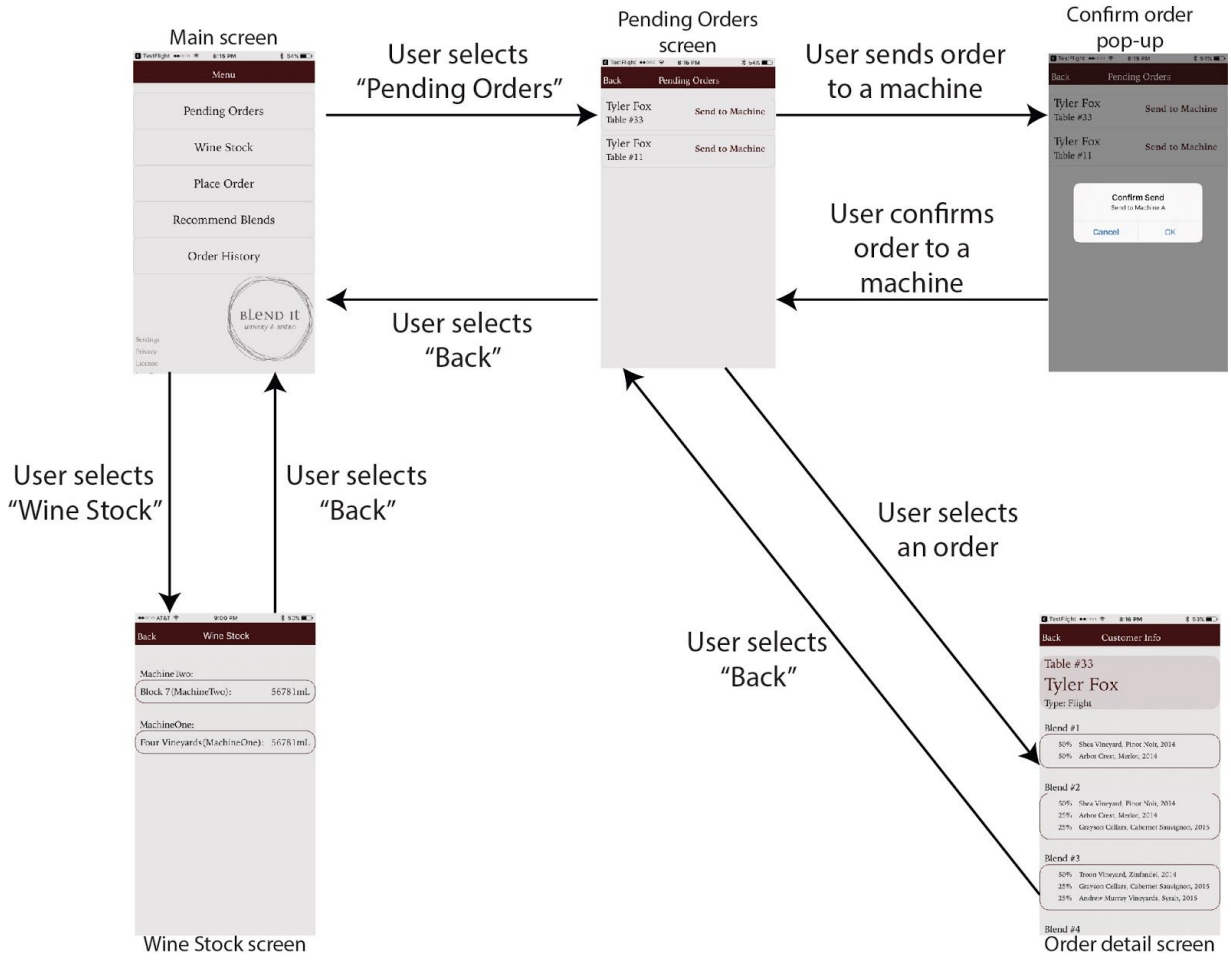


Figure 5: Basic bartender application flow

The above diagram details exactly which pages are implemented in the bartender application. The user starts at the menu page and after clicking the Pending Orders button, they are taken to a screen that shows all the customer orders that have not yet been fulfilled. If the Send to Machine button is pressed, then that order is pushed to the server to go to the Wine Blending Machine. If the user clicks on a customer's order, they are taken to a screen that gives more specifications about the order. Also from the main menu, the user can click the "Wine Stock" link and be taken to a page that displays information about which wines are present in which machines and the amount of each wine. All other pages are not implemented and only stubbed out.

System Integration and Testing

FMEA

The final FMEA can be found in the Appendix [6]. While analyzing what could cause our system to fail, we realized that many of the failure points are out of our control. The potential failures are either problems with the Wine Blending Machine, or problems with the network. We did some work with the Mechanical Engineering team in order to minimize potential failure and allow us to detect failure and recover successfully. One main failure mode we can control is the applications crashing. We can control this through testing and making sure the memory usage for the application is reasonable and well managed.

Design Verification Plan & Report

Item No	Specification	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES TESTED		TIMING	
						Quantity	Type	Start date	Finish date
1	Calculate remaining wine	Fill a barrel connected to the ME machine with water. Attempt to empty the barrel by ordering from the customer application.	If the machine does not produce a partial pouring as its last pour from the barrel and the machine does not signal the barrel's emptiness unless there is less than 10oz, than this test is passed.	ME team	Planning			4/15/2017	4/21/2017
2	BLEND IT mobile application usage	Test for Bartender or Customer application crashes by having a person use the application. This test will cover all screens and use cases.	Users do not experience crashes while using the Bartender or Customer application.	Bartender and Customer application teams	Execution	4	I	3/15/2017	3/21/2017
3	ME machine gets disconnected	Unplug ME machine to simulate loss of connection to the ME machine.	Bartenders are notified when a connection loss happens.	Bartender application and server teams	Planning	3	I	3/15/2017	3/21/2017

Table 4: DVP & R.

The DVP&R shown in Table 4 above details our primary testing plan. We have minimal access to the Wine Blending Machine and additional time and testing will be needed to verify that Item 1 meets the requirements. Item number 2, however, is currently in progress and will be for a while past the Senior Project class. We have set up a Google Form for feedback related to both the applications and the server. There is an Apple developer account that holds the beta versions of both applications as they exist now. Our client has access to this account and has added himself as well as several others so they can work with the applications on their phones and provide feedback.

Server Testing Architecture

1. **Code Pipeline**

We utilize AWS Code Pipeline in order to automate our deployments and perform continuous delivery. This pipeline is able to run all of our tests automatically before deploying the code to the production stack. When we commit to our git repository, AWS detects the change and automatically kicks off a build, deployment, and testing phase before any code reaches production.

2. **Unit Testing**

After each git commit the Code Pipeline runs a build automatically. During this stage, our unit tests are run. This ensures that each individual unit of code is functioning properly and meets expected standards. This also attempts to catch any problems as early as possible in the process. The unit tests are designed to be easily runnable on your personal machine before committing to git so you can make sure the build will still succeed.

3. **Integration testing**

After the build succeeds in the code pipeline, everything is deployed to a Beta stack. This stack mimics production as much as possible in order to allow us to test without contaminating data. Before promoting code to Production we run integration tests against Beta to make sure everything functions end to end as expected. This allows us to check for any holes there might be in the unit tests and make sure everything is wired as expected.

Overall System Analysis

Customer Application

Currently, the customer application fulfills its primary use case of enabling customers to order flights of blends. We implemented this use case with the Blend Wizard - a string of pages that guides users to customize their ‘flavor profile’ and order wines that fulfill that flavor profile. In the process of implementing this use case, we have created software infrastructure and graphic designs that will allow very quick development of the use cases that have not yet been implemented.

One important feature that was not fully implemented and tested was session management. We developed login state management and login user interfaces in parallel, but they were not connected or integrated into the application. This feature is something that will have to be implemented in the future. Since login has not been fully developed, neither has the customer profile page. This will have to be fully developed and connected to the server by a future team.

The feed page is another feature that will need continued development. The feed is partially implemented and tested, but does not meet all of its requirements. It currently presents a list of wines blends that are statically defined in the application. These wine blends show the friend that created it as well as an additional page detailing more information about the blend. When the feed is finished, it will present information on popular blends of wines, friends’ blends, and custom information submitted by BLEND IT employees as it appears in the server.

The customer application is robust. TypeScript has allowed us to avoid much of the risk inherent in JavaScript and React development. React’s componentization has also allowed us to avoid the lifecycle management problems of iOS and Android development. We expect the application to remain consistently unbreakable, even through feature addition.

Bartender Application

The bartender application serves as the last administrative stop before a customer's order is sent to the machine to be poured. Our requirements primarily revolved around the application being able to successfully pull information from the server to display on the application and being able to send information to the server in order to indicate that an order is ready to go to the machine. In this respect, we were successful in meeting our requirements. The application currently does both of those things, including dynamically displaying information based on which customer is selected. However, there are other secondary requirements that we did not meet like being able to place an order from the bartender application and having an order be placed in the order history page after it was fulfilled. These requirements were not met due to time and technology constraints. We had significant issues setting up the application in the chosen environment and successfully implementing simple pieces in our chosen language. After some training and initial hurdles it was much easier, but this did delay us.

In terms of inherent error, the bartender application does not have any errors present. The application has a low crash rate and performs as expected. There is little concern for the robustness of the bartender application.

There is room for future work on the bartender application. There are several pages that have some initial framework available, but are not fully implemented. The page that allows bartenders place an order will use the customer application's order process is not present. This requires the customer application to fully implement their Blend Wizard so that the bartender application can plug into it. The Order History page, while not implemented, has a sturdy framework available and requires only a server API and transferring UI from the Pending Orders page. Login with bartender IDs would also be a feature for future work. The core functionality of this application is implemented, but polishing and secondary requirements will be fulfilled during future work. The Recommend Blend page, while not a core requirement, is also stubbed out but needs finished implementation.

Server

The web server has made a lot of progress in its final quarter. At the end of last quarter, the web server was connected to the customer and bartender applications, however most of the JSON objects being sent to the application teams were static. Additionally, before this quarter the database lacked ample data that could be used for our integration testing. Finally, the web server was not fully connected with the wine blending machine prior to this quarter. For these reasons, the main goals of the web server centered around:

1. Placing a large amount of data into the tables of our database for testing. This allowed us to test our APIs and nail down our schema, which is crucial for the HTTP POSTs and GETs between the application teams and the wine blending machine.
2. Modify our AWS Lambda functions (JavaScript code that runs in the cloud) so that they are querying, modifying, adding, and deleting items from the tables in our database.
3. Get the wine blending machine fully connected to the web server in such a way that the web server was able to properly handle orders as they flowed from the customer application to the bartender application.
4. Develop an algorithm that takes a customer's selected flavor profile and generates blends that will match this flavor profile based off of what wines are currently on tap in the machine.
5. Leave the web server in such a way that it will be easy for someone to continue work on it after completion of this project. This will allow the web server to keep servicing the mobile applications as more features are added.

The web server completed goals one through three this quarter, with goal five being kept in mind throughout the entirety of the project. There is a large amount of test data in the tables of our database and the webserver is now modifying, adding, and deleting items for each API call. In addition, the wine

blending machine is connected to the web server in a way that will properly handle orders as they flow from the customer application, to the bartender application, and then to the machine. This is accomplished by using a FIFO queue, which guarantees orders to be delivered once and only once from the queue to the machine. Once the machine successfully pours an order, the machine sends a confirmation to the queue and then deletes the order from the queue. The Orders table and Wines table are then modified to reflect the changes in wine amount and the status of the order.

There is room for future work on the web server. As the bartender and customer applications continue to implement new features, the web server will need to continually provide the APIs needed to pull from the database. One of the big goals moving forward is the development and testing of an algorithm that turns a flavor profile into a list of blends for a customer to choose from. The idea is to implement a simple algorithm using weighted sums. The difficult part of the algorithm is testing it. To know if the algorithm is effective or not, we need a sommelier or wine expert to test the blends. Due to the difficulty of testing the algorithm we have not yet worked on implementing an accurate algorithm.

Another idea that will be key to the application's success is using a customer's session token to authenticate all interactions with the BLEND IT server. Without proper security measures, an attacker could decompile the code to get the API URLs and start making requests without authorization. This check for a valid session token will need to be added to all server-side logic.

Most of the infrastructure is in place for further application development, but more API functions need to be generated. In order to facilitate this, our automatic deployment pipeline for testing and deployments will allow simple iterations that produce quality APIs for the applications to leverage. We have also built our data schema in a way where we can expand as needed. We have templates laid out that allow our schema to easily be changed if needed. The careful consideration we have put into the overall architecture of the webserver will allow it to meet the many needs of both applications.

References

[R1] "AWS Overview." *AWS Overview - Getting Started with AWS*. Amazon, Inc., 2017. Web. 30 May 2017.

[R2] "Product Details." *Blendtique Wine Company:Build Your Custom Wine*. N.p., 2013. Web. 30 May 2017.

[R3] "Product Developers & Technology Consultants." *Unravelling the Secrets of Wine*. Cambridge Consultants, 2 Nov. 2016. Web. 30 May 2017.

[R4] "Protocols." *Protocols - AWS IoT*. Amazon, Inc., 2017. Web. 30 May 2017.

Appendix

[1] **Additional Team**

The BLEND IT Application Senior Project Team is working in conjunction with the BLEND IT Mechanical Engineering Team, which consists of three members:

Connor Clarry, ME

Matt Moren, CPE

Russell Temple, ME

[2] **Customer Requirements**

- A login page where the user has the ability to create an account or login as guest
- A winemaker's dashboard page where a user can do the following:
    - Blend wines
    - Redeem reward points
    - View reviews of other users blends
    - View what wines are currently available to blend with
- If a user selects blend wines they are taken to a screen where they select the type and amount of each wine they will blending

[3] **Decision Matrix - React Native**

Metric	**Weight**	**React Native**	**NativeScript**	**Cordova**	**Native iOS**
Cross Platform	.3	1	1	1	0
Performance	.15	.9	.9	0	1
Ease of Programming	.3	.9	.8	.8	.5
Community	.25	.8	0	.8	1
Total	1	.89	.66	.74	.55

Table 5: Decision Matrix for User Application Tools

[4] **Decision Matrix - AWS**

Metric	Weight	AWS	Microsoft Azure	Google Compute
Scalability	.2	1	1	1
Easily Serviced	.2	.9	.8	.7
Documentation	.2	1	.9	.9
Cost	.4	1	.7	.7
Total	1	.98	.82	.8

Table 6: Decision Matrix for Server Tools

[5] **Use Cases**

Use Case ID	1
Use Case Name	Order a flight
Created By	Brady Aiello, Alex Bartlett, Max Bendick, Aly Chapman, Tyler Fox, Luis Manjarrez, Cory Mayer, Rebecca McKinley, Brian Moore
Last Updated By	Brady Aiello, Alex Bartlett, Max Bendick, Aly Chapman, Tyler Fox, Luis Manjarrez, Cory Mayer, Rebecca McKinley, Brian Moore
Date Created	November 3rd, 2016
Date Last Updated	November 3rd, 2016
Actors	3. Customer 4. Web server
Description	Customer orders a flight of 5 blends
Preconditions	3. Customer has an active session 4. Wine chosen for blends is available
Postconditions	Bartender receives order
Normal Flow	9. Customer opens the initial blending process page from main page 10. Customer creates a tasting profile by moving sliders for how much of each flavor (nutty, caramel, fruity, floral, herbaceous, woody, earthy, spicy) they want

	11. Tasting profile created by the user is sent to the web server 12. Web server responds with 10 blends 13. Customer selects 5 blends to create a flight 14. Customer orders the flight 15. The order is submitted to the web server 16. Web server confirms that the order is submitted
Alternative Flows	1. Users hits back button on any page a. The previous screen is displayed with data still filled in 2. User hits cancel on any page a. User exits the blending wizard
Exceptions	1. Session becomes inactive a. User is prompted for verification before ordering 2. Server becomes unresponsive a. Error message is displayed b. User can retry or exit 3. Internet connectivity lost a. Error message is displayed b. User can retry or exit
Assumptions	Wine availability is maintained

Use Case ID	2
Use Case Name	Establish a session
Created By	Brady Aiello, Alex Bartlett, Max Bendick, Aly Chapman, Tyler Fox, Luis Manjarrez, Cory Mayer, Rebecca McKinley, Brian Moore
Last Updated By	Brady Aiello, Alex Bartlett, Max Bendick, Aly Chapman, Tyler Fox, Luis Manjarrez, Cory Mayer, Rebecca McKinley, Brian Moore
Date Created	November 3rd, 2016
Date Last Updated	November 3rd, 2016
Actors	1. Customer 2. Web server 3. Bartender/Employee
Description	Customer verifies age and their presence within the establishment with bartender
Preconditions	Customers are in the establishment
Postconditions	Active session is created
Normal Flow	1. Bartender selects customer seating location 2. Bartender application requests authorization token from web server 3. Bartender verifies ages of all customers at the table 4. Customers input authorization token in the application 5. Customer application sends authorization token to web server requesting a session 6. Web server responds with an active session
Alternative Flows	
Exceptions	1. Customer is not 21 a. Customer doesn't get an authorization token 2. Authorization token expires a. New token is requested 3. Authorization token is invalid a. Web server doesn't create an active session b. Error is displayed in customer application
Assumptions	

Use Case ID	3
Use Case Name	Bartender fulfills order
Created By	Brady Aiello, Alex Bartlett, Max Bendick, Aly Chapman, Connor Clarry, Tyler Fox, John Kraemer, Luis Manjarrez, Cory Mayer, Rebecca McKinley, Brian Moore, Matt Moren, Russell Temple
Last Updated By	Brady Aiello, Alex Bartlett, Max Bendick, Aly Chapman, Connor Clarry, Tyler Fox, John Kraemer, Luis Manjarrez, Cory Mayer, Rebecca McKinley, Brian Moore, Matt Moren, Russell Temple
Date Created	November 3rd, 2016
Date Last Updated	November 3rd, 2016
Actors	4. Blending machine 5. Web server 6. Bartender
Description	Bartender communicates with the blending machine to fulfill an order
Preconditions	3. Order is placed 4. Machine is active
Postconditions	Bartender fulfilled requested order
Normal Flow	10. Bartender selects order from bartender application 11. Order is sent to web server from bartender application 12. Web server sends the first pour to the machine 13. Bartender places glass on machine 14. Machine senses placement of glass and queues pour 15. Bartender confirms pour on machine screen 16. Machine dispenses pour and displays order being dispensed 17. Machine confirms pour completion with web server 18. Bartender removes glass 19. If there is another pour in the order web server repeats process starting at step 3 20. Bartender confirms order finished in bartender application
Alternative Flows	
Exceptions	1. Human error results in a failed pour

	a. Bartender click re-pour instead of finish b. Bartender can select the pour to re-pour 2. Machine error (Empty keg, valve failure, etc.) a. Bartender clicks re-pour instead of finish b. Bartender can select the pour to re-pour c. If machine is still failing follow troubleshooting guide
Assumptions	

Use Case ID	4
Use Case Name	Customer saves a blend to their profile
Created By	Aly Chapman
Last Updated By	Aly Chapman
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	Customer
Description	Customer saves a blend
Preconditions	1. Customer has already ordered and tasted a flight 2. Customer has an account
Postconditions	Customer has saved a blend
Normal Flow	1. User navigates to recently ordered flight 2. User selects a blend to save 3. System saves the blend
Alternative Flows	2b.Pop up notifies user the blend is already saved 2c.Pop up notifies user that there was an error trying to save the blend
Exceptions	1. Communication error between application and server causes save to fail (Alternative 2c)
Assumptions	

Use Case ID	5
Use Case Name	Rate a blend
Created By	Max Bendick
Last Updated By	Max Bendick
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	Customer
Description	Customer rates a blend
Preconditions	Customer is on the detail page of this blend
Postconditions	Any time a blend's rating is displayed, the given rating is shown.
Normal Flow	1. User taps a star rating on the rating graphic. 2. Given star rating is highlighted.
Alternative Flows	1. Popup notification tells user of the failure.
Exceptions	Network communication to post the rating fails (alternative flow 1).
Assumptions	

Use Case ID	6
Use Case Name	Discover a blend
Created By	Cory Mayer
Last Updated By	Cory Mayer
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	User, Customers, Bartender
Description	Customer discovers a wine blend via the social feed.
Preconditions	People have posted wines to social feed, bartender has posted recommendations.
Postconditions	Wines posted by customers are displayed in social feed.
Normal Flow	1. User selects “Feed” page 2. User is presented with wines posted by customers and bartenders. 3. User selects wine shown on feed. a. User is able to view information about the blend on “Wine Profile Page”. i. User is able to rate blend and leave reviews. ii. User is able to recommend blend to friends 4. User selects back button to return to feed.
Alternative Flows	1. User has deleted blend. a. Presented with nice error. 2. Blend has no reviews. a. Text indicates that there are no reviews. 3. Blends fail to load. a. Previously retrieved blends are shown and nice error message occurs. 4. Review fails to post. a. User will receive nice error message and allowed to try again or close.
Exceptions	1. User has deleted blend (alt flow 1). 2. Loss of network connectivity (alt flow 3). 3. Review fails to post (alt flow 4).

Assumptions	User has a full account.

Use Case ID	7
Use Case Name	Share a saved blend
Created By	Rebecca Mckinley
Last Updated By	Rebecca Mckinley
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	Customer
Description	Customer shares their blend on social media
Preconditions	Customer successfully created and labeled a custom blend, customer is logged in via a social media site like Facebook or Twitter
Postconditions	Customer successfully posted about their blend on social media, the customer is at the dashboard page
Normal Flow	1. Customer clicks the page/button that indicates they want to post on social media 2. Dialog window opens with a link to the blend and a description auto filled into the post box 3. Customer can add additional comments about blend in the dialog box 4. Customer clicks the “post” button and their post is pushed to social media
Alternative Flows	1. Customer clicks “cancel” during post a. Social media post will not post and dialog box exits
Exceptions	Login session on social media site expires
Assumptions	Assuming valid connection and login information for chosen social media site

Use Case ID	8
Use Case Name	Check the wine levels
Created By	Luis Manjarrez
Last Updated By	Luis Manjarrez
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	Bartender
Description	Bartender will check to the wine levels in the reserves
Preconditions	Wine tanks exists and are connected to a machine, wine is accessible from the tanks
Postconditions	Wine levels are displayed to the Bartender
Normal Flow	1. Bartender clicks on “Check Wine Levels” 2. All of the Wine Levels are displayed on the screen in alphabetical order 3. Bartender can set the setting in which the Wine Levels are displayed
Alternative Flows	1. Wines aren’t accessible to the Bartender application
Exceptions	Bartender loses access to server
Assumptions	Wine levels are recorded in the server database

Use Case ID	9
Use Case Name	Update the Wines
Created By	Tyler Fox
Last Updated By	Tyler Fox
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	Bartender
Description	The bartender updates what wines are currently in the machine
Preconditions	There is information in the database about the new wine(s) being added
Postconditions	The web server is updated with the new wine(s) that are now in the machine
Normal Flow	1. The bartender or an administrator logs in to a webpage to interact with server 2. Bartender/Administrator updates the database with the new wine(s) that are going into the machine
Alternative Flows	
Exceptions	1. A wine tank runs out unexpectedly a. The bartender is notified of this error
Assumptions	

Use Case ID	10
Use Case Name	Bartender place an Order
Created By	Brady Aiello
Last Updated By	Brady Aiello
Date Created	November 8th, 2016
Date Last Updated	November 8th, 2016
Actors	Bartender, Backend, Mixing Machine
Description	An active bartender places a blend order to be fulfilled by a local mixing machine.
Preconditions	1. Bartender must be logged in to an active Bartending Session. 2. There must be a database that accepts and maintains customer orders. 3. The bartenders logged in to an active session must have access to a subset of these orders, depending on establishment, serviced tag, and primacy.
Postconditions	1. An order confirmed, and forwarded to one of the establishment's mixing machines.
Normal Flow	1. The bartender has already logged in to the bartender application. She views a queue of all pending orders. 2. The top item of the queue will be sent to the machine. Next to each item, there will be 2 buttons: 1) Instantly finds a machine to fulfill the order, 2) Lets the bartender manually select a machine. The bartender presses the first button. 3. The bartender application sends a direct message to the target mixing machine containing the blend request. 4. The mixing machine performs the request. 5. Upon completion, the mixing machine sends a callback to the application's backend, marking the order as "serviced". 6. The bartender can manually remove items that are marked as "serviced", in the case of an error.
Alternative Flows	1. Instead of letting the application decide where to send the request, the server specifies a machine to fulfill the order.

	2. All other steps follow as before.
Exceptions	1. If there are no longer any machines that can fulfill a particular order, then a new order will need to be made. 2. If the machine intended for the request runs out of one of the wines necessary to complete it, the bartender application should be notified. The bartender application should always be notified when a tank has a low level for any wine.
Assumptions	1. The bartender is in an active bartending session. 2. The customer is present and over 21. 3. Payment is handled through a separate mechanism and interface.

FAILURE MODE AND EFFECTS ANALYSIS

Item:	BlendIt Application and Server	Responsibility:	All	FMEA number:	1
Model:	Current	Prepared by:	All	Page :	1 of 1
Core Team:	Blendit App Team			FMEA Date (Orig)	1/17/2017 Rev: 1

Process Function	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Class	Potential Cause(s)/ Mechanism(s) of Failure	Occur	Current Process Controls	Detec	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results				
												Actions Taken	Sev	Occ	Det	RPN
Calculate remaining wine	No wine remaining, server thinks there is more than 10 oz remaining	Loss of order	5		Lost packets	5	Low rate of DB transactions	8	200	Recommend volume sensor on-machine	ME team, never	none	0	0	0	0
Calculate remaining wine	More than 10oz remaining, server thinks there is none	Waste wine	2		Lost packets	5	Low rate of DB transactions	8	80	Recommend volume sensor on-machine	ME team, never	none	0	0	0	0
ME machine gets disconnected	Ethernet connection is lost	Orders cannot be sent to the machine	9		Wire gets pulled out	2	Plug it back in	1	18	none	never	none	0	0	0	0
Blend It mobile app usage	Application crashes	Loss of data in memory	5		Large memory usage, violation of native contracts	3	Restart app	2	30	Reduce memory footprint	Bartender and Customer app teams, March 2017	none				0