Team Piggy
Taylor Bauer, Ryan Pierce, Mitch Simmons, Tommy Nguyen, Megana Chinachaiagari
Team 6
2/10/2020

Project Name

Piggy

Project Synopsis

Productivity app for professional bartenders and bar managers to manage and easily update bar inventory, quickly make and retrieve cocktail recipes, and automatically generate a pricing model and quantities for liquor ordering.

Project Description

We are undertaking this project because local bar owners have expressed frustration with the current landscape of behind-the-bar management apps. While there are a variety of liquor inventory apps on the market, few do it with a professional bartending context in mind: they are missing features like automatic order reporting and quick updating of current stock. Many at-home cocktail apps have recipe functions but are not robust (or, more importantly, fast) enough to be realistically deployed during a busy shift behind a bar.

As a group, we are uniquely positioned to develop this application. Two of our members are currently employed as bartenders locally, and we plan on leveraging their proximity to local bar owners and managers throughout development to seek feedback and ideas about the project. Another member has extensive experience in retail management and will have valuable insight into the needs of users who will be tracking inventory with our app.

Piggy will implement an algorithm that pulls from past inventory management data to provide the users with a pricing model and inventory reorder model that would maximize profit. The algorithm will help to determine what to charge for drinks based on their sales, which will be determined by comparing the amount sold to the amount ordered in the previous week. The algorithm will also provide other business analytics.

By the end of the project, we plan to deliver a working iPad app that has functionality to track and easily update liquor inventory, create and quickly search for cocktail recipes, and automatically generate liquor orders based on current inventory and user-defined pars.

Project Milestones

See Gantt Charts (figure 7) at end of document for more detail.

Our original milestones focused around the planning of the actual project, as well as the planning for an estimated timeline. As we have progressed through an entire semester with weekly TA meetings, our visions for the project were adjusted according to suggestions, individual workloads, as well as key features we found to be beneficial for an effective product.

Our first milestones focused on confirming and identifying the Swift and SwiftUI languages to be the main learning area necessary to create an IPad or Iphone application. We have successfully identified our project details - including its functionality, the components to be broken up amongst our team members, and a new timeline that is best suited for the time remaining. As of now, we have designed ER design models for the databases we will be using. Our back end and front end charts were designed and finalized in October. Our current largest milestone is being able to create a simple demo by the second week of February.

In regards to the UI, we are in the process of implementing features like the Recipe tab and Menu aspects. Overall, the UI should be planned to be built out by early March. At the same time, now that the BSAN algorithm is created and being finalized, we hope to finish implementation of the database by the end of February and the algorithm by early March.

A final creation of art and other aspects to create a likable experience for the user will be one of our last milestones, towards the end of March, as we wrap the project up. By the first week of April, we plan to have a working application that we can create a demo video for and present and potentially even deploy.

Project Budget

Our original budget included Mac rentals for two of our group members since we all needed access to Xcode for development with Swift and Swift UI (three members already owned a MacBook or an iMac). We were instead able to rent a Mac Mini and a MacBook Air from the KU, so we didn't have to pay for monthly Mac rentals which ultimately saved us about $40-$50 a month per Mac. We also no longer had to consider developing on a different OS platform since everyone had access to the Mac OS.

We were also able to save the $20-$50 we originally budgeted for visual work and logo design since the art is all being done "in house" by Ryan. The only expense we still need to consider is the $100 in developer fees required to put our app on the App Store should we decide to do so later on.

Work Plan

Taylor - Project lead. Design and UI lead. Building inventory menus and system, recipes system, and BSAN algorithm implementation.

Ryan - Recipe creation UI, recipe display UI, and assisting with the recipe database. Also in charge of creating the art used in the app.

Tommy - Recipe and Inventory UI implementation

Mitch - Business Analytic Development, BSAN algorithm implementation, Database design and Implementation

Megana - Back end development with Database design and implementation

Final Project Design

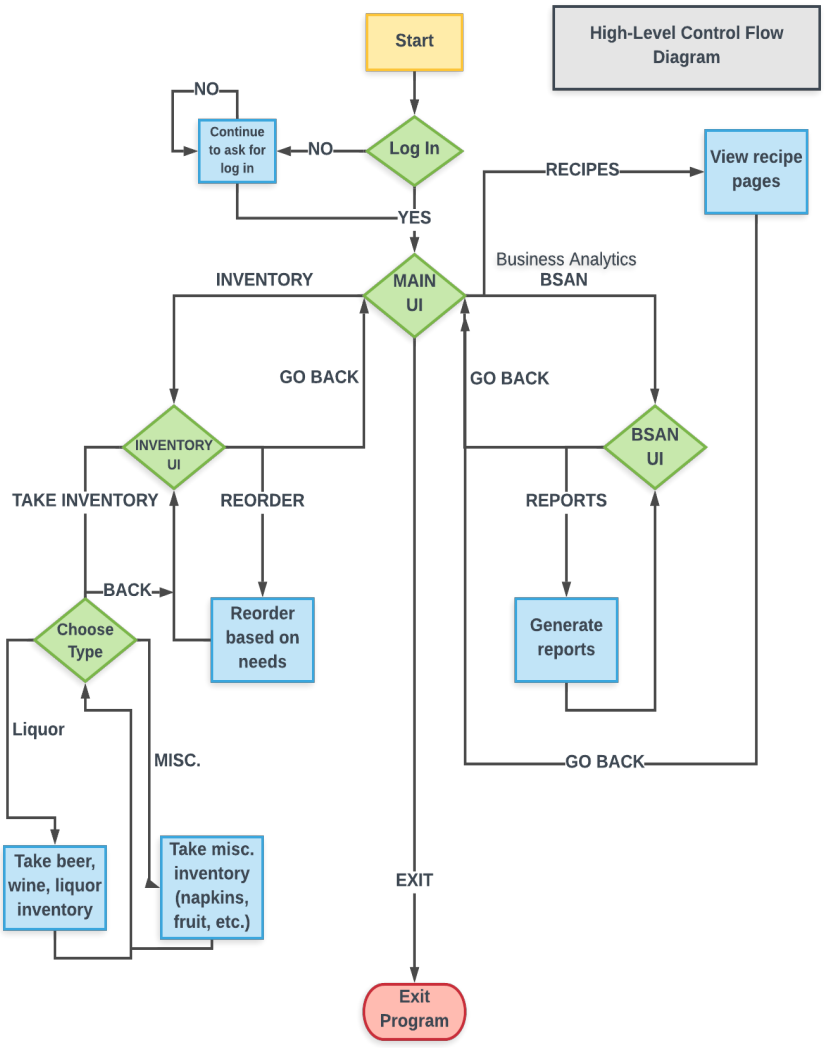

Fig 1. High-level control flow diagram

Problems and our app's features that solve them - Bars face a unique set of challenges that set them apart from other food service establishments. Unlike restaurants, most of the inventory is non-perishable. The rate at which a bar goes through different items in an inventory is inconsistent at best, but wildly unpredictable at worst. As such, in our research, most busy bars take a complete inventory count at least twice a week. However, the evolution from pen-and-paper inventory management into the digital age has not been a smooth one, and many establishments are still using what are now outdated and arcane methods of inventory management that involve a lot of repetition of boilerplate tasks (think: updating an Excel spreadsheet) and, critically, a significant amount of guesswork in ordering and pricing.

Piggy solves all of these problems without interrupting the already established workflow that bar owners and bartenders have been using for years. The tentpole feature of the application is an extremely streamlined yet flexible inventory counting and reporting interface. Piggy gets rid of the need for spreadsheets and mental math by providing the user with a quick way of updating inventory and, most importantly, an automatic report generation feature that will tell the user exactly how much of each item they should order based on user-defined pars and current inventory. Additionally, Piggy features customizable grouping of items to enhance workflow when performing the count--items that are physically close together can be easily rearranged on the list to create a more linear path for the counter and avoid the need to go back and forth between shelves.

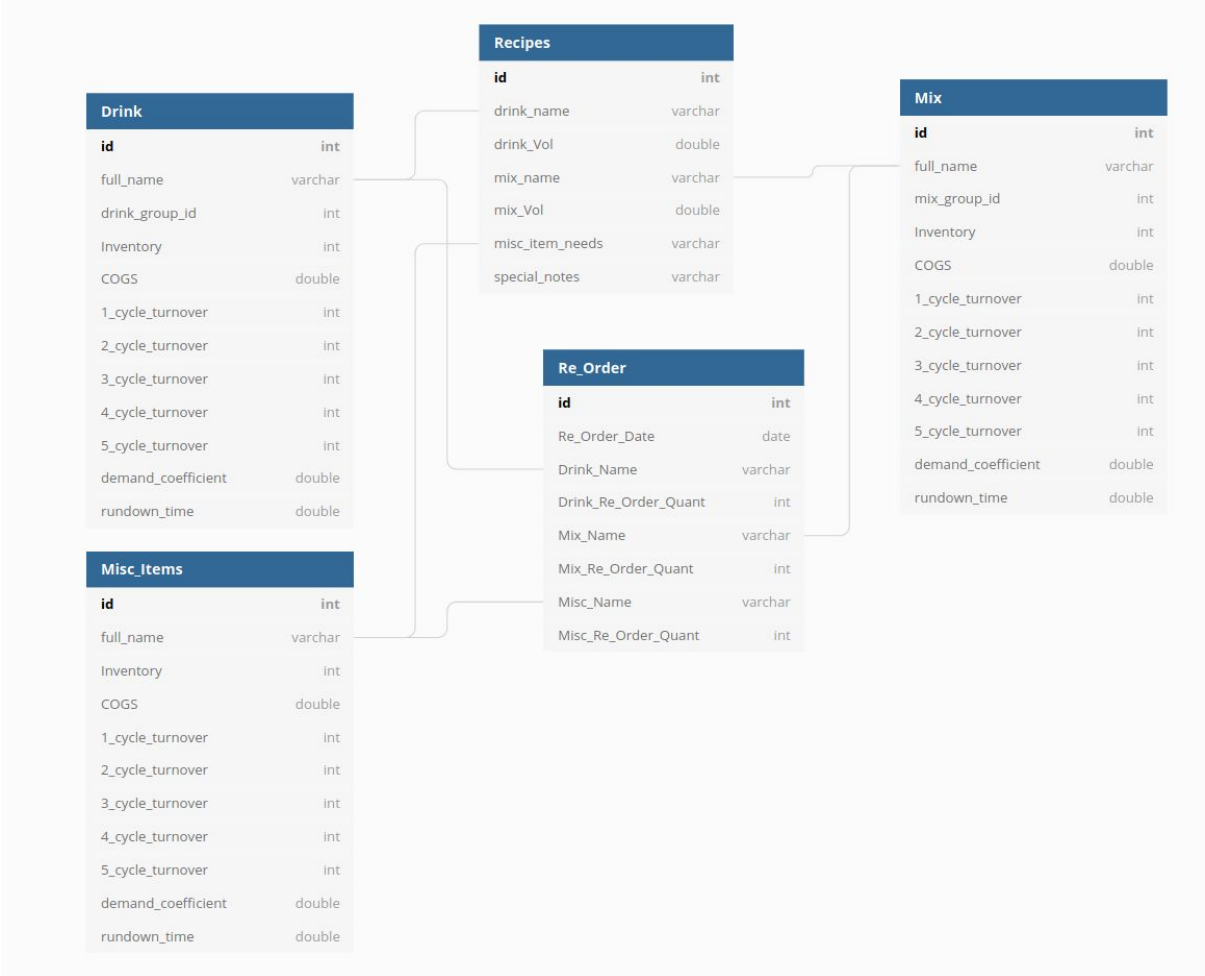

Fig. 2 - ER Diagram

Bartenders we spoke to also expressed frustration with the process of procuring a specific cocktail recipe in the business of a rush. Most bars seem to use recipe cars stored in a small filing cabinet or rolodex. The reported problems with these systems are fourfold: it is slow to find a recipe, the cards can go missing, recipes can get outdated as inventory or other workflow changes, and the cards are difficult to read in a dark bar setting. Because many bars already have iPads for other purposes behind the bar, it is a natural fit for Piggy to integrate a cocktail recipe storage and retrieval system that solves all four problems. The app makes it quick and easy to rapidly input recipes and allows for immediate, legible, and up-to-date retrieval of recipes. Since we will already have a database full of liquors and spirits, we can use this information to generate recipes based on what the user has stocked. If, say, the user has a list of ingredients for a cocktail, but is not exactly sure how those ingredients could be used together, our app will be able to output possible recipes based on the user's current inventory. If the user doesn't have all of the items for a single cocktail, it can instead make suggestions for recipes that the user is only missing a few ingredients for so that they know what they need. Recipes can also be stored for later use, which will be helpful in the bar setting and a faster alternative to flipping through index cards.

Another problem that many bar owners and managers face is pricing of cocktails. In our research (and experience) cocktails are priced on the price "shelf" upon which the primary spirit in the cocktail sits (well, call, premium, super premium, etc.). However, the actual cost of a pour of a cocktail can vary wildly depending on factors like the amount of said primary spirit is used and other ingredients, such as other spirits or things like citrus juice. Additionally, the variance of prices in just one "shelf" can be rather large: one bar we spoke to had "call" spirits that were $18/bottle and others that were as high as $35/bottle--almost twice as expensive to the bar but being sold at the same price to the customer. How are you supposed to price a cocktail when there are so many factors involved?

Piggy tackles this problem by providing the user with a simple interface that automatically calculates the cost of a pour of a cocktail or other beverage including beer and wine based on the recipe. This feature really ties the different corners of the app together, and combines the recipe and inventory features in a unique way that bar owners and managers can utilize to help determine how much they should be charging for each beverage. It is the feature in Piggy that we are most excited about, because, in our research, we have not seen it executed well and know it can be a major selling point of Piggy.

Piggy will rely on an algorithm using data collected during the inventory process. The algorithm will consist of calculations derived from demand data, supply data, price changes, and the changing rates of demand. The data will be collected through the manual inventory process, weekly compilations of data, and weekly relational data to create a pricing model that will maximize profit for professionals, or minimize cost to amateur bartenders.

Utilized Technologies - Piggy is developed in the SwiftUI framework in the Swift programming language using the XCode suite. SwiftUI is a new framework from Apple that uses declarative syntax to describe UI elements and their functionality. It is designed to make it easier to develop for iOS, macOS, and iPad OS at the same time, but for the time being we are reducing our scope to just an iPad app until we learn more about what porting between platforms actually entails. For our backend, we will be using SQLite3, which will be more than powerful enough for our purposes of managing relational databases of inventory and recipes. SQLite requires minimum configuration and allows for manipulation of local database files, which suits our needs because we will not be using any sort of cloud storage.

Design Constraints

Technical: From the beginning of the project, we knew we wanted to develop an application that can easily fit into bars' long-established methods of doing inventory and streamline the process, not reinvent their systems from the ground up. In our research, most bars do already inventory management on an iPad, so we are constrained to that platform. Due to the nature of our application being on the iPad platform, we are restricted to using Swift as our programming language. This constraint is a complicating factor, because none of the five of us have any experience working in Swift or even in Objective C, Swift's predecessor. Certain aspects of Swift will actually make this transition very easy (reference-counting garbage

collection means we won't need to worry too much about memory management), but other elements, such as the new SwiftUI suite in XCode, have a steep learning curve that we are already combatting. Additionally, the restrictive Apple ecosystem requires that iPad applications are developed on Mac OS, a constraint that will be addressed in our business constraints section. This programming language constraint naturally begets another technical restraint, which limits our platforms to the Apple family of products. Luckily, with Apple's new Mac Catalyst program, it should be easier to port Piggy to other Apple platforms (such as MacOS) after developing our iPad OS app, but a port to any other platform would require a ground-up rebuild. The specific framework we're constrained to is, as mentioned above, is SwiftUI, which, although user-friendly, is relatively new on the market and doesn't have as many extensive tutorials as other potential frameworks. While this makes development more difficult, we are excited to have the experience of working with a cutting-edge framework.

Business: Many of our business constraints are due to the nature of our project being developed in a yearlong senior design capstone course. The schedule is rigidly set by the professor: we are to be done with the planning phase by the end of this semester, and need to deliver a working product by the end of the Spring 2020 semester. Fortunately for us, beyond the cost of tuition for us as individuals, we do not anticipate running up against any budgetary constraints (largely thanks to the fact that none of us are paid): KU's EECS department has been more than willing to help pay for things like Mac rentals for those of us who need them. Team composition is similarly set in stone for us, as all five of us need to contribute significant work in order to receive a passing grade. Finally, we are not anticipating to have to worry about any specific software licensing issues, as we are building our application from the ground-up.

Piggy: Inventory Main Screen Mock-up

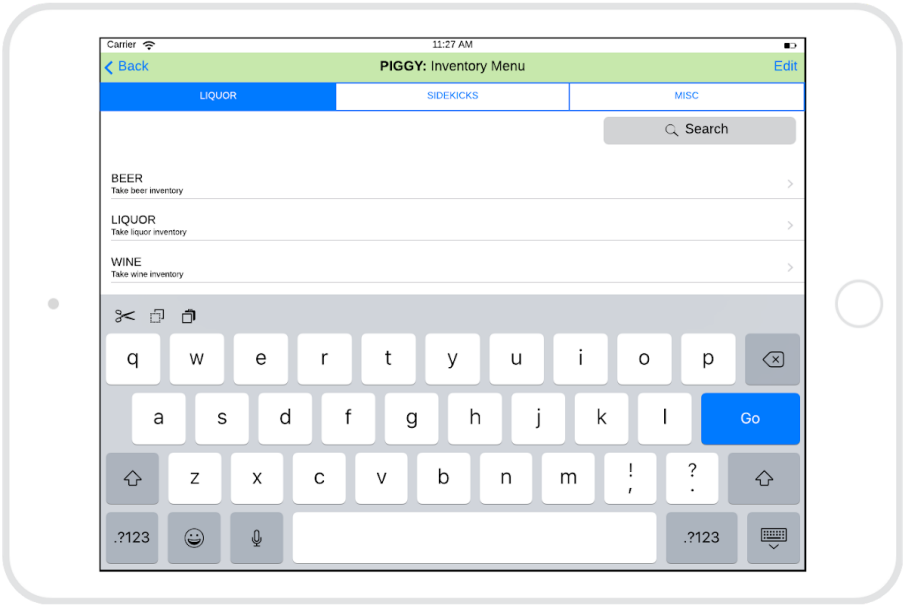

Fig. 3 - Early UI Mockup

Fig 4 - Main Menu and Liquor Inventory counting interface

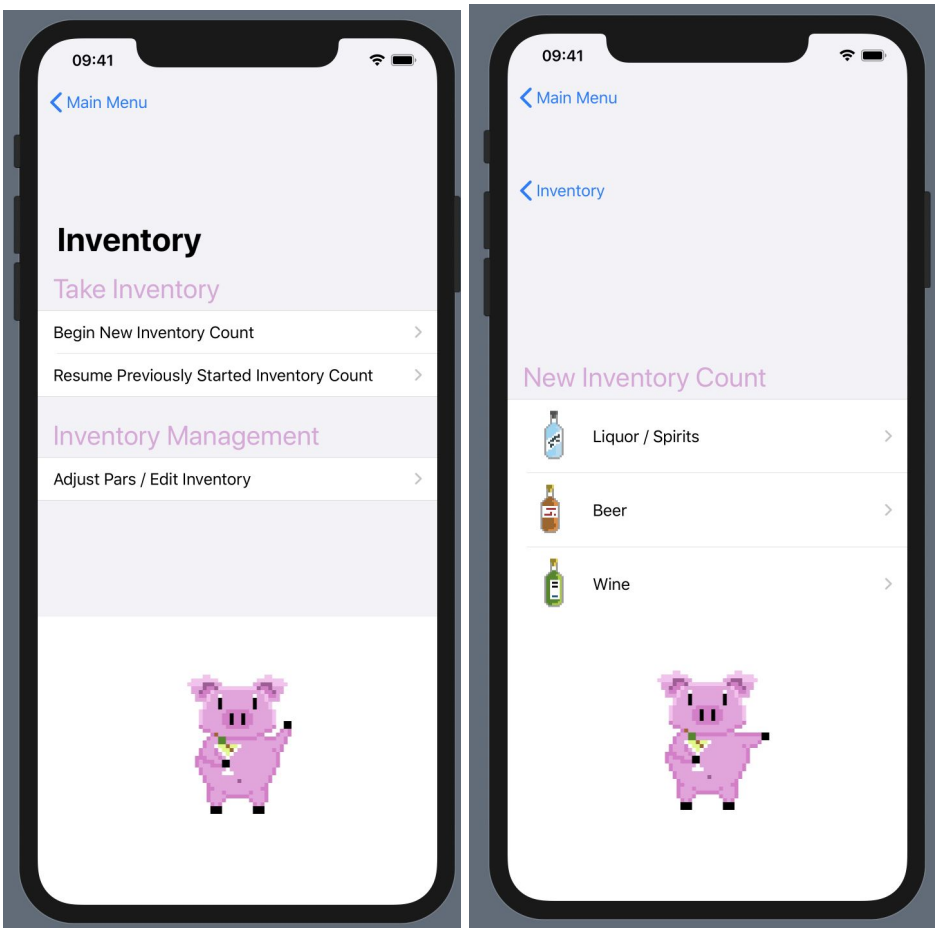

Fig. 5 - Inventory count workflow

Ethical Issues

Disallowing creation of recipes with high alcohol content - A major component of cocktail design is creating a drink that customers find sufficiently strong and enjoyable without serving too much alcohol. For example, it would be risky if a customer orders their usual quantity of cocktails at a night out only to find that they are sick because the cocktails they consumed contained a significantly higher alcohol content than they are accustomed to. As app developers, we are able to help prevent this dangerous behavior by requiring each recipe to contain no more than 2.5 ounces of 80 proof spirit (or equivalent serving) before being saved to the app's database. This amount is about 125% the amount of alcohol in a standard cocktail, and about as much as can be found in the strongest martinis. This way, we as developers, can help ensure that our app is being used safely and ethically when being used by bartenders.

Intellectual Property Issues

Not using brands that are not our IP in app creation and marketing - While, almost certainly, individual bars and bar owners will be entering the exact names of spirits, including

brand names, into their database, these brands and brand names are not our intellectual property and, as such, we will be using generic product names, like “apéritif” or “well vodka” in lieu of specific product names, like “Campari” or “Smirnoff”, respectively, in all app development and future marketing. It will hopefully be clear enough to customers that each category of product is completely customizable, and, as such, they can later add the specific product names in themselves. This way, we can give the customers full control of their inventory categories while avoiding use of intellectual property that doesn’t belong to us.

Change log

- **Reducing our scope and dropping the iOS app** - The decision to keep ourselves to just one platform (iPad OS) was made because of our slower-than-expected progress in the first few weeks of the semester. However, because we will be using SwiftUI, which debuted since the beginning of the semester, there is a strong likelihood that we will find porting to iOS to be less difficult than the initial plan outlined.
- **Addition of supply/demand algorithm** - Preliminary designs for the algorithm have been created. The algorithm may change as implementation begins and our constraints on available data become more apparent.
- **Updated work plan -** As the project went along, our individual roles came better into focus. The work plan section is now more task-specific.
- **Changed Ethical issue** - Per our teacher, we are changed our ethical issue from the one we had last semester
- **Changed Intellectual issue** - See above
- **Gantt chart update** - As with the work plan, several lines/tasks were added to our gantt chart as the project progressed
- **Included final UI screenshots** - To replace the mockups that were in the previous report
- **Updated Database ER Diagrams** - To keep up with changes in the algorithm and interactions between tables
- **Updated project budget** - We initially budgeted for several items that we didn’t end up needing to pay for, so the updated version reflects those changes.
- **Updated Project Description** - Added some more brief information about how the business analytics will be utilized in the application.
- **Update “app features”** - Added new information about the recipe creation/calculation feature of our app.
- **Changed Milestones -** The original report did not have concrete milestones as per our timeline. This has now been updated above.

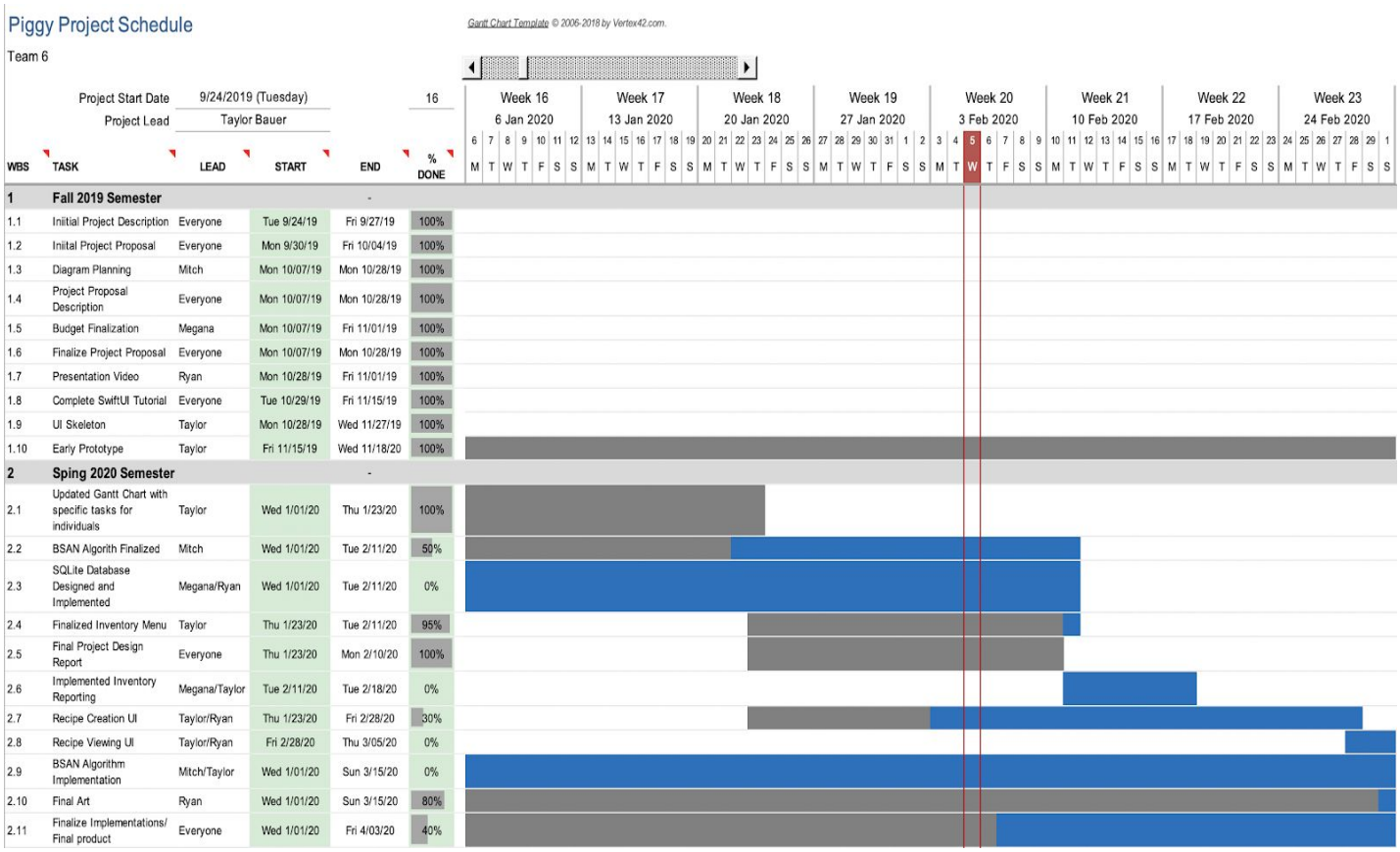

Fig. 7 - Gantt Charts (updated as of February 5, 2020)