Team 3: Final Project Design

Andre Kurait, Natahn Nichols, Daniel Zolotor, Rob Nickel, Thor Lyche, Alex Kunz

Project Name: Line Hopper

Logo:

Project synopsis: Allow students to order and pay for KU Dining menu items through a mobile application to reduce lines at on-campus dining locations.

Project Description: The motivation behind this project stems from the overall wait times at KU dining facilities, which leads to customer dissatisfaction and reduced sales for KU. This project is especially relevant now that KU dining has implemented declining balance meal plans. This will allow KU students to have greater freedom to use their dining plan at any of the dining facilities on campus, instead of primarily the dining halls. Several team members noticed that many students do not use facilities like the JayBreak Cafe because they don't have time to wait in line for their food to be made in between classes. By implementing a mobile application, students would be able to order while in their class, pay online, then pick up their food in between classes. This would lead to increased sales for KU and a more pleasant experience for KU students.

The end result for this project will be a mobile application prototype that would include the ability to order food items from specific KU dining facilities and pay for them remotely. Once we have proven the concept through our prototype, we could hand off our project to KU Dining's IT team, and let them integrate it into their current system.

Project Milestones:

First semester:

1. Collection of KU dining locations and prices (10/18/19)
2. Design mockups and user workflows (11/1/19)
3. Set up barebones backend AWS services with item retrieval and ordering (11/15/19)
4. Barebones front end with checkout, basic ordering (12/6/19)

Second Semester:

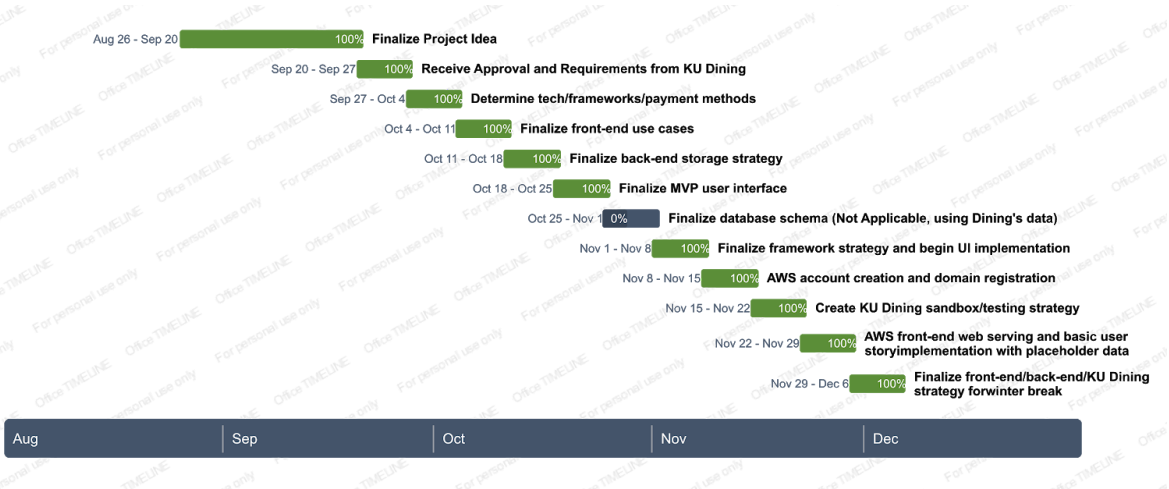
5. Integrate backend data with frontend UI (2/28/20)
6. Integration with a internet-enabled receipt printer for order creation (3/20/20)
7. Demoable order completion (4/24/20)

Budget:

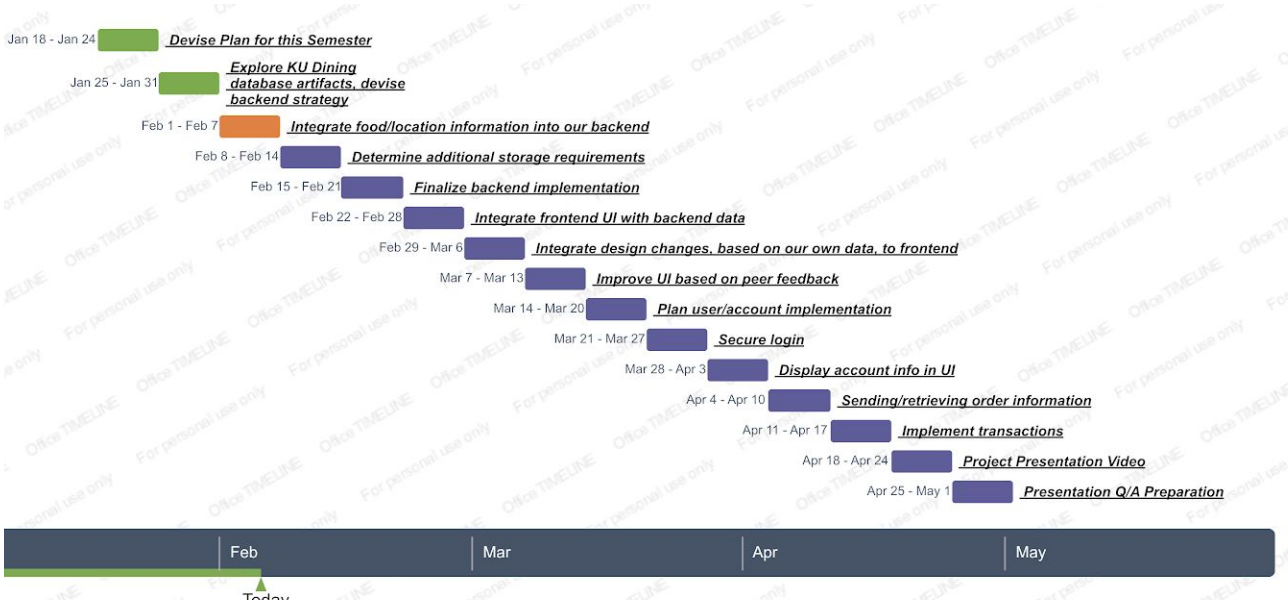
Wireless Receipt Label Printer	$151.99
TOTAL	$151.99

Work Plan: Alex Kunz and Thor Lyche will work on the front end development of the application as well as UI/user experience. Daniel Zolotor, Nathan Nichols, Rob Nickel and Andre Kurait will be architecting and implementing the AWS backend for efficient and cost-effective ordering using a micro-service architecture.

First Semester Gantt Chart:



Second Semester Gantt Chart (as of 2/10/20):



Preliminary Project Design:

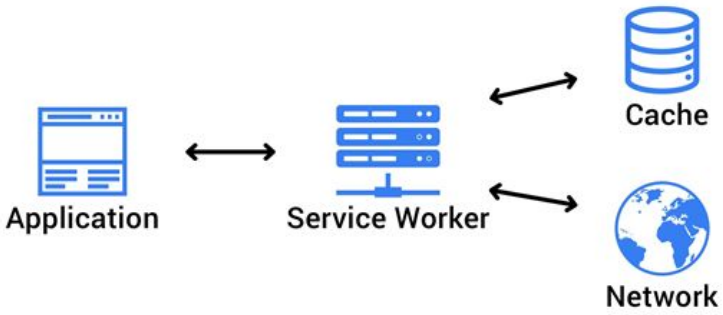
User experience: Here is our expected user story for someone who would like to make an online purchase. First, the user will visit our website on any device. If they are on a mobile device they will be prompted to add the site to their home screen for ease of use. Afterwards, they will be greeted with a list of KU Dining locations to choose from. Once a location is chosen, we will be able to retrieve up to date stock, order fulfilment delay, and pricing for that location. Our user will then select the items they'd like to order, as well as the pick up time. This pickup time can be as soon as possible, or a future time. After this, the user will be prompted to confirm and checkout. They can enter their payment information to use a normal credit/debit card or they can login to their KU account to use dining dollars. At this point the retail location's label printer will print the order letting the cashiers know that a mobile order has been placed. When the order has been fulfilled, the user will get a notification that their order is ready. Users will then be able to walk up to the location, verify that their order is completed on the screen, and pick up their order.

Progressive Web App: We are using Angular, the JavaScript front-end framework, to create a progressive web app. A progressive web app is a website that looks and feels like a native mobile app. Since it is a website, users don't have to sift through the app store to run our app; all they need is their browser. This will make our app accessible to the largest amount of students.

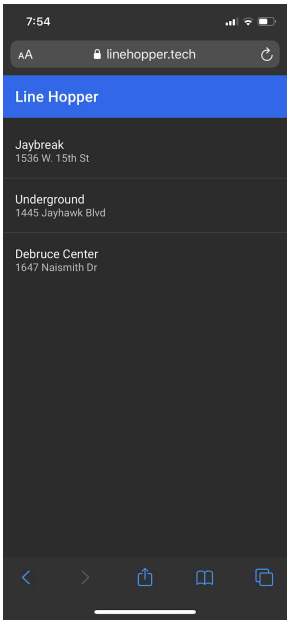
Progressive web apps also allow us to implement key features of native mobile apps to provide an optimal user experience. First, our app will implement offline caching to ensure our app is fast and reliable. Second, we will keep orders fresh by notifying users of order updates with

push notifications. Lastly, we will increase usage by creating a downloadable homescreen icon. Our app will make mobile transactions efficient and convenient.

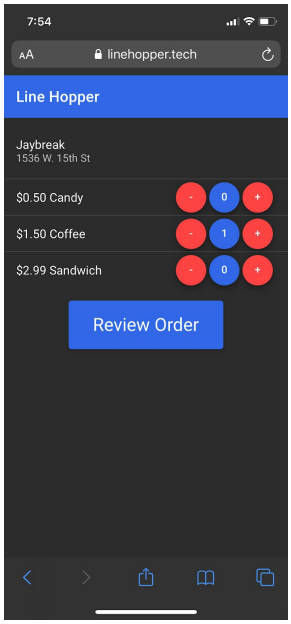
We will implement some of our core functionality with service workers. A service worker is a file that is run in the background of a browser that can listen for and respond to network requests. This allows our app to provide content while offline, which has lots of benefits. Our app can update its cached content without having to open the app, which means the user will have relevant data loaded when they return to the app. Also, push notifications are handled by service workers notifying the user of a network update. Service workers allow our app to be as responsive as possible.



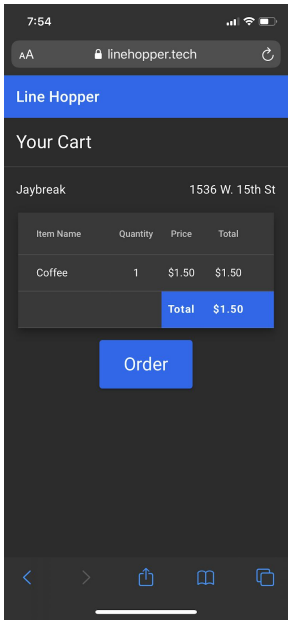
User Interface: The basic user interface and functionality is shown below. Currently it only supports hard-coded locations/menus, but these will be replaced with our backend's data soon.



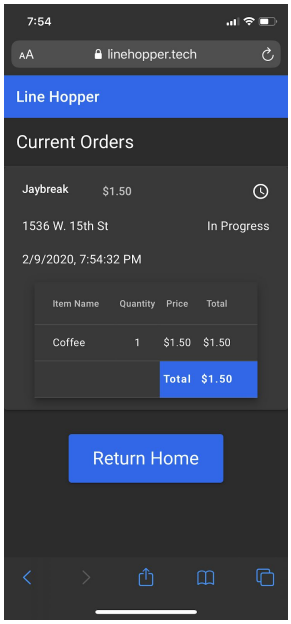
1. Landing page, displays location options



2. Page for a specific location's menu



3. Cart for a specific location



4. Current/active orders

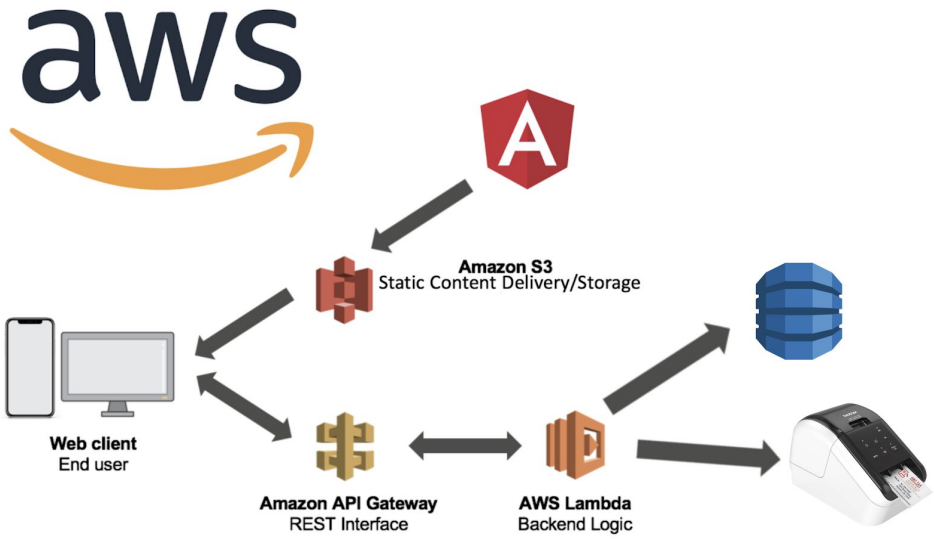
KU Dining Integration: We are creating all data manually from online resources.

Click here for location names, menus, and hours

KU Dining maintains a database with a table for each register's transactions. Our app will be treated as a register with our own table. A column within that table will indicate a single register at that location to represent where the payment would have gone to. At the end of each day they will be able to add up all the transactions as normal by looking at each transaction in our table.

Credit Card Purchases: We will use the payment processing service named Square for credit card purchases. With the API they provide, we can simply pass the credit card information that the user provides. Again, we will never store this information reducing our liability and decreasing severity of any security breaches.

Backend: AWS (Amazon Web Services) will be our cloud provider. We will use the S3, API Gateway, DynamoDB, and Lambda services in order to connect all of our services and frameworks. The Angular web app will be served with an Amazon S3 container. This is possible because the compiler for angular is included within the code, which allows our program to be statically generated. Using an Amazon S3 container will allow our app to be easily deployable and available to all of our consumers. Our web clients will then make REST API calls as needed to AWS's API Gateway. These API calls will then be redirected to our Lambdas. Our lambdas, which will be powered by Golang, will connect our app with our Non-Relational Database and our printer to notify the barista of the order. Then, our backend will have access to all of the resources necessary to provide our content. We will be able to gather and return all dynamic data, as well as create and publish orders. Using Lambdas will provide two major benefits. First, we will remain secure and require authentication for payments. Second, it will offer a seamless and hassle free experience for requests that don't require security, such as searching for menu items and pricing information. Our AWS infrastructure will be created in the Serverless infrastructure in order to provide transferability, verification, and code history.



Business Constraints: Our primary business constraint is scheduling. All of us have lots of other classes, assignments, and activities to attend to, which makes it difficult to plan meetings. This is not like a full-time position, where our sole focus is the software we are developing. We were not given timely access to use useful technologies, so we must create our own, which takes more time. We will have to make communication a priority with them to avoid setbacks. Software licensing might be troublesome if we want to take full ownership of our app while using KU software. This is expanded upon in the intellectual property section. When it comes to our budget, we should not have any issues. AWS Free Tier will suffice, and our other budget requests will be handled by the given senior design budget.

Technical Constraints: One of our major technical constraints is that our app is highly dependent on KU Dining's tech stack. We must manufacture data and build everything with integration in mind, since we won't be able to test with their services along the way. A constraint for our front-end is working around KU Dining's request for our app to be as low maintenance as possible; this will make it easier for them to integrate it into their system after we graduate. We chose the Angular framework because it was created around backwards compatibility and it was lightweight. Because of this, polyfills will not be a problem as Angular continues to release new versions, and it should be easy for KU IT's staff to maintain.

Ethical Issues: There are two main ethical concerns our project may encounter, one physical ethical violation and one online potential violation. Because our app will allow customers to order online, the handling of their personal financial information (credit card, debit card, etc) will be in our hands. We will need to take measures to ensure that their information stays protected and secure. We are now planning to use an outside payment platform such as Square to handle all of our payment processing. Even if we keep the data secure on our end, there is always the chance that Square could have a data breach. We may end up processing our payments through KU's already implemented system (Oracle) for ease of use. The second ethical dilemma we may encounter is students stealing the food/ beverages paid for by other students. As it currently stands, after a student orders their food the cashier will immediately make the food and hand it to them. Under our system, the food could be ready and waiting on the counter to be picked up by the person who placed the order. This may raise some concern if students begin to take orders that were not placed by them. This could be resolved by having students show their order confirmation before they are able to receive their food or drink.

Intellectual Property: One intellectual property constraint we see is working with/ integrating into KU's systems. Because we will be running transactions and working with KU dining, we may be told that we need to integrate our code with theirs. This may become an issue if our code becomes an integral part of their system, and they then refuse to grant us rights to our own software. There are two ways we will get around this constraint. First, we can give up rights to certain pieces of code that serve no functionality other than connecting our project to KU's currently available system. Second, if we are going to be integrating more specific code into KU's systems we will create contracts with them in order to classify which code is ours and which is theirs. This allows us to retain IP rights to the majority of our project.

Change Log: Red highlight indicates removal, green indicates addition

- **Project milestones:**
    - First Semester:
        - Before: Barebones front end with secure login, checkout, basic ordering (12/6/19)
          After: Barebones front end with checkout, basic ordering (12/6/19)
          Why: we will no longer support secure KU login due to KU IT's lack of resources.

    - Second Semester:
        1. Before: More fleshed out app with order history, dining location selection, visual UI elements, and completion notifications. (2/28/19)
           After: Integrate backend data with frontend UI. (2/28/19)
           Why: More relevant goal based off of our current progress.

        2. Before: Setup of secure communication channels back and forth from our servers to KU Dining's backend (4/24/19)
           After: Demoable order completion (4/24/19)
           Why: We are no longer planning on working directly with KU dining's database due to their slow timetable.
- **Budget:**
    - Before:

Receipt Label Printer	$50
LG 24' LED 720p Monitor	$100
Intel Compute Stick	$150
TOTAL	$300

    - After:

Wireless Receipt Label Printer	$151.99
TOTAL	$151.99

    - Why: Due to an increased separation with KU IT in this stage of the project, we needed to upgrade to a wireless printer as we would not have the permission to use a wired printer on the network. Additionally, we decided to take out the intel compute stick and monitor as they were unnecessary in this stage of the project.

- **Work Plan:**
    - Before: “Our current plan is for Nathan Nichols and Daniel Zolotor to work on back end payment processing including connecting to Oracle Micros Simphony as well as overall project architecture. Alex Kunz … UI/user experience. Rob Nickel and Andre Kurait will …”
    - After: “Alex Kunz … UI/user experience. Daniel Zolotor, Nathan Nichols, Rob Nickel and Andre Kurait will …”
    - Why: Shifting responsibilities to account with the change in the deliverables.
- **Preliminary Project Design**:
    - The user experience section was moved within the “Preliminary Project Design” section for clarity. Also, we removed the statement that stated that students could login to their KU account because we will no longer support that feature.
    - The “Frontend” section was retitled to “Progressive Web App” to more accurately represent its purpose.
    - The “User Interface” section was added below the “Progressive Web App” section to display the UI progress.
    - **KU Dining Integration**:
        - We removed all references to KU’s database and the CBORD system because we are no longer directly accessing KU’s database or working with KU card systems due to lack of resources. We included our new backend strategy that would fit in well with KU IT’s infrastructure.
    - **Backend:**
        - We removed references to Oracle and CBORD while stating that our lambdas will connect with our own database created with DynamoDB. We have to create our own database because we won’t be able to access KU’s.
    - **Business Constraints**:
        - Before: “...we are developing. Beyond our team, we will have to work around KU Dining/IT’s schedule for anything that involves them. This could delay our development if we are not given timely access to necessary technologies. We will …”
        - After: “...we are developing. We were not given timely access to use useful technologies, so we must create our own, which takes more time. We will …”
        - Why: We are acknowledging that we were not given ample resources to fully integrate with KU IT, and because of this we will require more time to create our own technologies from scratch.
    - **Technical Constraints**:
        - Before: **“**… our app is highly dependent on KU Dining’s tech stack. This means CBORD Card System compliance is mandatory. We will also need to integrate our app with Oracle Hospitality’s backend, because Oracle will provide the API to populate different components of the app. …”

    - After: **“**… our app is highly dependent on KU Dining’s tech stack. We must manufacture data and build everything with integration in mind, since we won’t be able to test with their services along the way …”
    - Why: To align with the decreased development with KU IT while still keeping integration at the forefront of our design to allow for future extensibility.
- **Ethical Issues**:
    - We removed the statement “both their KU information (if using their KU dining accounts)” because we are no longer supporting KU dining meal plans.