University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2020

ServeSmart

Reddington Walters, Chase Pareti, Ethan Brugger, Spencer Heald, Mitchell Merrick, Dylan Huber, and Nicholas Coluccino

Abstract

Many restaurant point-of-sale (POS) systems are inefficient and confusing. We aim to amend this with our design of a portable, minimalistic, and complete restaurant management software: ServeSmart. Our platform is built for tablets as well as stationary terminals and will handle tasks, tables, pending orders, payment, inventory, and logistics. This was implemented in React Native so that it is deployable on all target devices.

Our platform will have numerous benefits for establishments that use it. Firstly, the wait staff will be able to decrease turnaround time by having an easier way to organize and complete their tasks. In turn, this increases customer satisfaction, gratuity, and wait times at the door. Sales data from each working shift will be stored, organized, and analyzed to allow management to track their metrics and make calculated business approaches. The end goal is to maximize efficiency and profit for both the entire restaurant team.

1.0 Problem

As stated, we claim that a disorganized and inefficient method of handling orders when using restaurant software is a problem. Customer satisfaction should be centered around providing the best service. Thus, restaurants should not have to fund expensive software or rely on veteran employees who are familiar with the current system.

A robust restaurant business can have new employees up to speed in the least amount of time possible as well. Additionally, current POS systems stem from stationary terminals. Oftentimes when a shift gets busy, all terminals will be in use and workforce individuals must wait to access them.

Without a fix to these problems above, restaurants lack the ability to maximize service speed, guest satisfaction, and profit.

2.0 Objective

Our objective is to create a mobile and stationary POS software platform to increase efficiency, profit, and guest satisfaction among restaurants. Utilizing our system will decrease turnaround time for the wait staff, simplify orders and extra instructions for the kitchen to reduce error, and maintain checkouts and reports to provide data analytics for management. Our business is to speed up tasks for wait staff in the food industry (which will save time and increase their tips), save money with an affordable all-in-one platform, and enhance guest satisfaction with a new and unique approach to food service.

3.0 Background

3.1 Key Concepts

Key technologies related to the problem:

- Front-end
    - There will be an interface for the workforce designed to be displayed on mobile and terminal screens. This will be where the wait staff can input and send orders to the kitchen, view and complete their tasks, and handle payment or extra requests.
- Kitchen
    - There will be a printer in the kitchen that prints orders on a neat and easy to read ticket when they are sent from the floor (the floor, in terms, is the patron area of a restaurant).
- Database
    - Our database will store employee and management login information, data for inventory and sales, outstanding orders and tasks, and restaurant metrics. The database is the primary tool for management because from this data they can predict labor hours for their schedules, keep track of inventory, and formulate relevant orders from the distribution trucks. It provides real-time data to the wait staff, which gives them the information they need as well as the ability to sign in and resume work on a different device. Using Google Firebase gives access to a real-time cloud server as well.

3.2 Related Work

Our team has extensive experience with software development and foodservice operations. Listed below are problems we experienced that exist with POS implementations across several different establishments:

1. Ease of use - staff members often need weeks to memorize the flow of the interface. Aside from the extensive training they already receive on the menu alone and service

techniques specific to the location they work at - this is redundant. Our aim is to resolve this by creating a system that is easy to learn, update, and customize.

2. Organization - service tasks are often expected to be ordered and executed by memory. Our platform aims to keep the POS at the staff's fingertips at all times as well as take their FOH (front of house) and BOH (back of house) tasks and manage them without needing additional input. This feature is experimental but designed in the hope to give staff the ability to focus more on the small details and deliver excellent service.

3. User interface - Having a clean UI allows wait staff to use the platform effectively to complete their orders/jobs.

Naturally, as the foodservice business has been in the works for a very long time, there are existing competitors in this project area. Many POS software platforms are for sale online, however, most are not suited for mobile devices. A native iPad platform, ShopKeep, is closely related with their UI/UX design, inventory management, and ease of accounting approach.

What sets our business apart is the versatile tablets and task ordering algorithm. Our aim is to wrap up the processes for wait staff and management as well as optimize the efficiency of the workforce.

4.0 Design

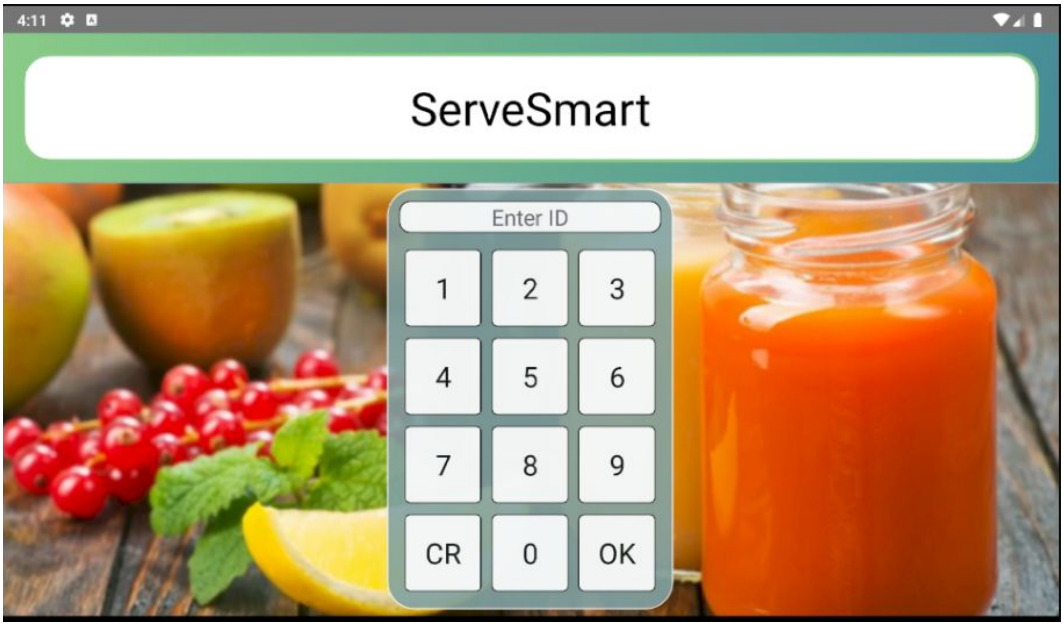
4.1 Requirements and/or Use Cases and/or Design Goals

- Staff are able to view and edit orders from the interface on a tablet or terminal
- The interface displays tasks and informational tooltips on their assigned section on the floor
- To implement our algorithm for task ordering in the most efficient order possible.
- The layout of the restaurant floor can be built and arranged by management
- The kitchen is able to view orders on tickets
- Staff are notified when food is completed by the kitchen and drinks by the bar
- Clean, simple, and friendly interface
- New menu items, specials, and promotions can be displayed for better use and sales tactics
- Data and metrics are gathered to be used by management to optimize their business

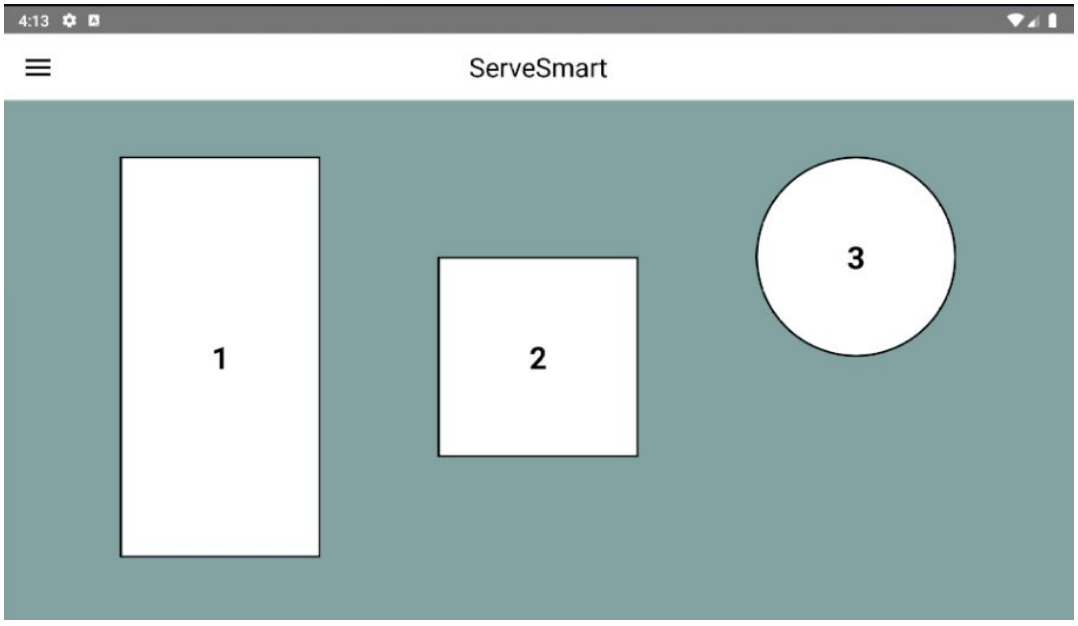
4.2 Detailed Architecture

ServeSmart uses React Native for the user interface and has Google Firebase operating as its NoSQL database. The intent for tablet usage is a simple Android device, thus Android Studio was used for emulating a development environment. Following are samples of the screens that were seen to completion.

The login page allows authorized staff and management to login with their PIN. Should the user enter an unrecognized PIN, they are unable to do so. A successful login will send the user to the “My Tables” page.



The “My Tables” page shows a layout of the restaurant floor. For brevity, an actual layout was not created and the floor consists of just 3 tables. Selecting a table will bring you to the current order that it contains or will create one if it doesn’t exist. By tapping on the menu icon in the top left, the drawer navigator will overlay and redirect the user to other features in the platform.

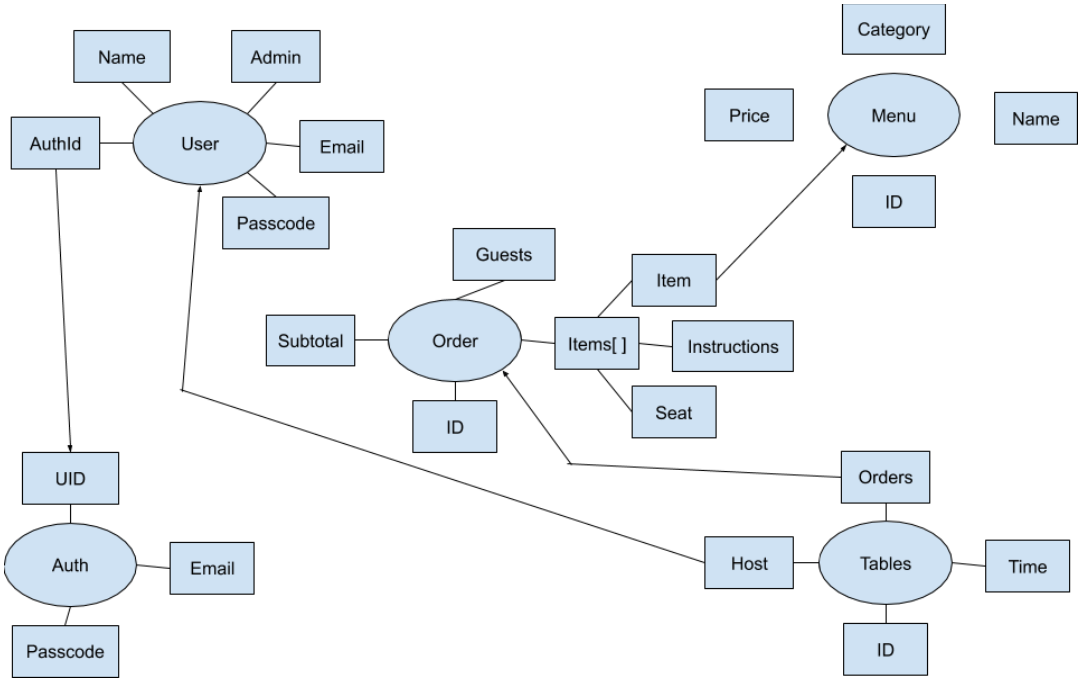


The drawer navigator, featured below, presents the staff with a way to access all the features of the app. Currently, the only features under development are accessing and modifying tables, viewing tasks, and looking at menu specials.

The order screen below is shown after selecting a respective table. In this example, the table has three guests, each with their own items, and a few shared items for the table. On the right side lies the menu, various quick controls, and categories containing menu selections.

For example, if the “Beverages” category were to be selected, the list of beverage items on the menu would then be displayed, Choosing one would add it to the selected seat. If a seat is not selected, then a new one will be created containing the selection.

Our Database Schema:



ServeSmart was always built and used in Android studio using an emulator. We put some research into ideal tablets for ServeSmart and had found that something similar to the specifications of the Asus ZenPad S 8.0 would work very well due to its low price, long battery life, and enough internal storage.

Implementation of ServeSmart went mostly as planned except for a few features. We put these features into the future work section and we were not able to implement them due to Covid-19 taking away in-person meetings and development time from our developers. These are important features we wanted to have for our application as we believed they enhance the user experience and the overall functionality of ServeSmart. Unfortunately there was not enough time to complete said features.

Throughout ServeSmart's development, the team learned many valuable lessons. We learned that even among a team of 7, complex features take a lot of time. We also learned that setting a deadline ahead of schedule may get the feature finished, but not polished.

4.3 Risks

Risk	Risk Reduction
Miscommunication between wait staff and kitchen	Creating clear prompts for both wait staff and kitchen on the process of each order
Competitors and existing platforms	Prove our platform is an advantage over existing competitors at a reasonable cost
Damage to tablets	To prevent the tablets from being dropped, they will be harnessed to the user's apron with a safety chord. Restaurants can invest in cases to protect from water and scratch damage as well
Battery life and dependability	Tablets will remain in house at a charging station at the end of each shift. In the case of battery shortage during a shift, restaurants will possess a few extras

4.4 Tasks

1. Understand/gain background about POS systems and what we can expand upon.
2. Design a database and preliminary code for client & wait staff to get communication up and running.
3. Test the client and server connection and debug if necessary.
4. Implementat main client code and functionality with server code to back it up.
5. Work on front-end functionality and create common test cases that should be covered.
6. Test front-end functionality and make sure no bugs exist.
7. Create a smooth, easy to follow UI and develop a better user experience.
8. Test functionality by using test cases and having real people use the system.
9. Document our work and create a report and user manual for the system.

4.5 Schedule

Tasks	Description	Dates
First Sprint meeting	Meet together and determine the architecture of our program. Decide how to store, handle, and transfer data between client and server. Assign roles to each team member.	Meet together and determine the architecture of our program. Decide how to store, handle, and transfer data between client and server. Assign roles to each team member.
Begin work on Sprint 1	Each member works on their specific tasks for sprint 1. Front end members initialize the Native framework and back end members set up Firebase.	1/13/2020- 1/20/2020
Sprint 2 meeting	Begin enhancing and building upon database API's and the user interface. Start to create a logical flow with basic communication through the application's components.	1/20/2020
Begin work on Sprint 2	Each member does their assigned sprint tasks and reflects their progress through the Trello task board. Tasks can be added or removed as necessary.	1/21/2020- 2/5/2020
Preliminary Testing	Test client server communications. Ensure data is being properly sent through the application. Search for bugs and help each other with road blocks.	2/6/2020
Sprint 3 meeting	Take results from testing and re-plan the future structure and direction of the app. Argue for implementing new	2/7/2020

	features and removing unnecessary ones.	
Sprint 3	Each member does their assigned sprint tasks and reflects their progress through the Trello task board.	2/8/2020- 2/22/2020
Sprint 4 meeting	Discuss how to connect all different components and continue with working and creating each necessary component.	2/23/2020
Sprint 4	Each member continues their respective work and works to the best of their ability, asking and offering help when necessary	2/24/2020- 3/5/2020
Sprint 5 Meeting	Meet and figure out how each team-member is progressing in their development and come up with strategies to help improve development	3/6/2020
Sprint 5	Continuing the good development practice of updating trill and committing code after every session to ensure visibility to all team-members	3/7/2020-3/21/2020
Sprint 6 Meeting	Lay-out what is necessary to finish for the app's functionality	3/21/2020
Sprint 6	Start work on the final features needed for our app	3/22/2020-4/3/2020
Sprint 7 Meeting	Meet and discuss the process of the final features and the realistic outcome our developers can achieve	4/3/2020

Sprint 7	Finish remaining features	4/4/2020-4/15/2020
Clean-up Sprint	Implement final touches on styling and organization.	4/16/2020-4/20-2020
Testing & Debugging	Run extensive end-to-end tests to ensure functionality. Reach out to restaurants and receive feedback on our app	4/21/2020-5/1/2020
Documentation	Document our work with a report and a user-manual for the system.	5/1/2020-Finish

4.6 Deliverables

- Product presentation video: video overview of product
    - Raw video file
    - Unlisted youtube video
- Database schema: Logical view of the entire database
- User Interface: JS code for the employee interface
    - Github links to repositories
- Final Report

5.0 Key Personnel

Spencer Heald – Heald is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Database Management Systems, Programming Paradigms, and Software Engineering. He has worked as a software engineer to develop fruit budgets for the Agriculture Department at the University of Arkansas and worked in the food service industry throughout high school. He helped with the design of the front end and began to implement data analytics.

Reddington Walters — Walters is a senior in Computer Engineering at the University of Arkansas's Computer Science and Computer Engineering Department. He has completed relevant courses such as Database Management Systems, Programming Paradigms, Software Engineering, Operating Systems, and Computer Networks. He has been selected to be part of the internship program with J.B. Hunt during Summer 2020. He helped the development of the database and the necessary API's to connect the client and server.

Chase Pareti – Pareti is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He recently completed an application development internship with J.B. Hunt Transport Inc., has a full time position lined up for the summer of 2020, and also works as an independent contractor for CopiedCode: a local startup tech company in NWA. Additionally, he has years of prior experience in the foodservice industry and wishes to amend the problems he experienced with his skills. His primary

responsibility was the development and design of the user interface, structuring the platform's task ordering algorithm, and ensuring the platform meets the needs for both management and workforce at any restaurant.

Ethan Brugger – Brugger is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses Database Management Systems, Programming Paradigms, and Software Engineering, Computer Networks, Artificial Intelligence, Embedded Systems and Data Mining. He is currently working in a research lab for Dr. Jia Di in preparation for attending graduate school and also has relevant internships at Fortune 500 companies like Sam's Club, Walmart China, and J.B. Hunt in Software Engineering roles. His primary responsibility was development of the API that connects the UI and the database, while also offloading some of the computation time that these low powered tablets are burdened with.

Dylan Huber – Huber is a senior Computer Science major in the CSCE department and a junior Finance major in the Finance department of the UofA. He has completed relevant courses. He has had internships with multiple departments including Deposit Operations and IT. He was responsible for part of the front-end UI.

Nic Coluccino – Coluccino is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses in Database Management Systems, Programming Paradigms, Software Engineering, Computer Networks and Algorithms. He will be finishing his degree in the Spring of 2020. He owns his own software company, CopiedCode, comprised of college engineers and based in NWA. He recently completed a cloud engineering internship with ConocoPhillips. He was responsible for backend APIs and data handling.

Mitchell Merrick – Merrick is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses including Database Management Systems, Programming Paradigms, Software Engineering, Computer Networks and Data Mining. He has completed a software engineering internship with Cerner, worked as a software engineer to develop fruit budgets for the Agriculture Department at the University of Arkansas, and has accepted a software engineering position with J.B. Hunt for after graduation. He was responsible for the back-end and part of the front-end.

6.0 Facilities and Equipment

- Tablets for wait staff
    - Use for the main interface to submit orders and check process
    - Charging station necessary to be packaged together
- A restaurant establishment
    - Our platform is intended for these facilities
- Restaurant layout
    - The layout of the restaurant floor must first be mapped to be displayed to wait staff on their tablet
- Menus

- All data displayed on the tablet will need to be formulated from specific restaurant menus

7.0 References

[1] "POS Software: IPad Point of Sale Solution." *ShopKeep*, https://www.shopkeep.com/pos-software.