Abby Scarry and Chase LaRocca

CPST 4710: IT Capstone

Tulane University

Spring 2025

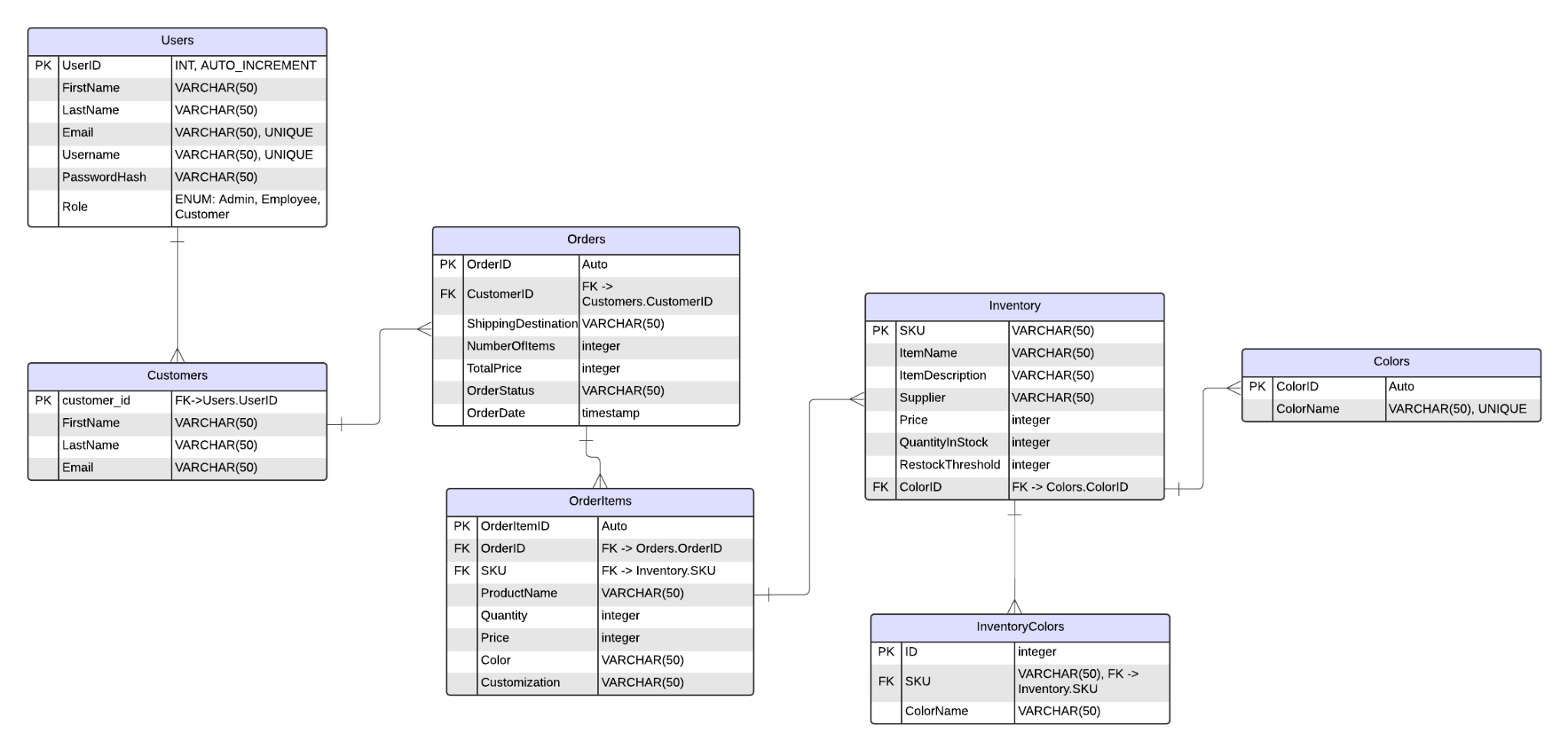
# Final Assessment: Written Overview

As the AppDev team, the goal of our project was to improve South Balance’s sales order process to meet Cybersecurity Maturity Model Certification (CMMC) Level 1 requirements. This is because South Balance is expanding into government contracting by partnering with the Army and Air Force Exchange Service (AAFES), which requires vendors to meet federal cybersecurity standards. To support this business objective, our AppDev team developed a web-based ordering application that replaces the company’s outdated manual order-taking system and adheres to CMMC level 1 requirements.

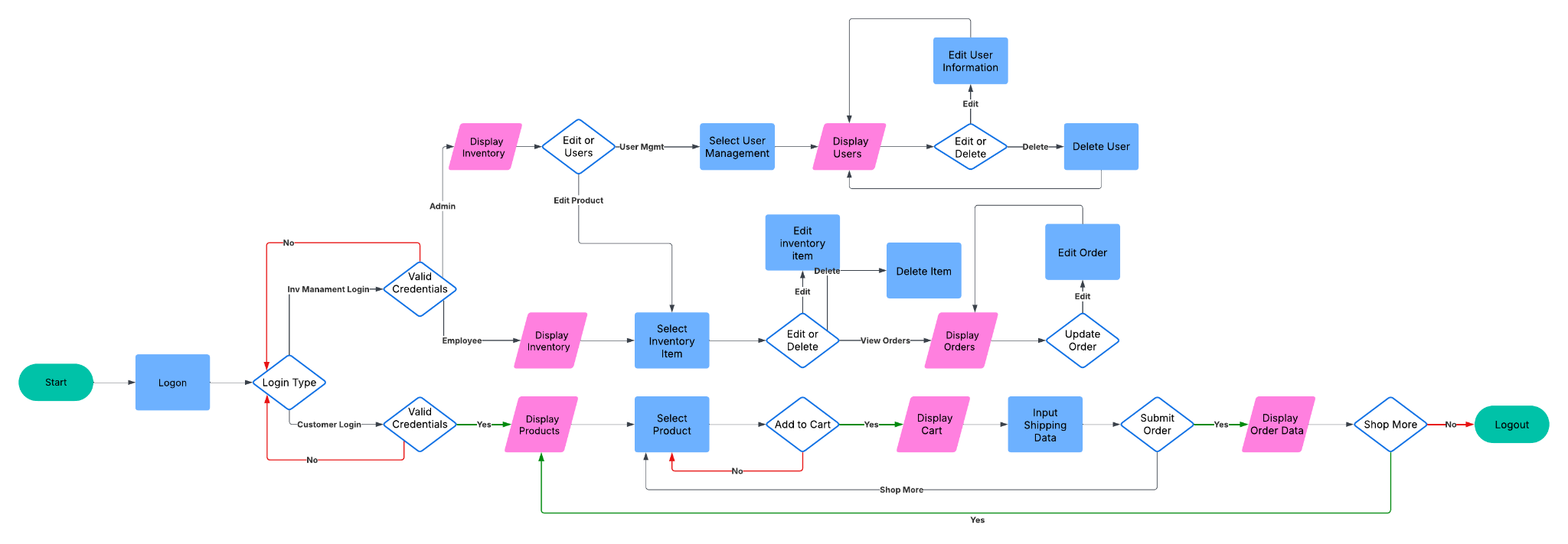
South Balance currently relies on spreadsheets, phone calls, and emails to place and track orders. This process is slow and leaves more room for errors in accuracy and security. By transitioning to an application, order-taking operations become more automated, efficient, scalable, and secure. The MVP version of our system currently supports only AAFES orders, but it has been designed with future expansion in mind, eventually allowing for use with all customers. To reach these goals, we worked together on planning out and completing each phase of the project step by step. We started by figuring out the system requirements and looking at how we could solve South Balance’s business objectives, while also making sure we built in the right security features to meet CMMC standards.

## **Development Roadmap**

Our development journey for this South Balance project has followed a clear step-by-step roadmap, where each phase built on the progress of the last. We began with some foundational work, such as learning about the CMMC Level 1 requirements, identifying project deliverables, and assessing risks. This helped us understand that the project’s core goal was not just building an ordering system but creating one that met strict CMMC standards. During these early weeks, we also defined the Minimum Viable Product (MVP) and drafted diagrams like our Entity Relationship Diagram (ERD) and user flow to visualize how the system would work. After this, we turned to work on the database design. These diagrams were important when it came to planning how our database would function. The ERD, pictured below, was useful as a guide and reference because it shows the relationships between the tables in the system, including Users, Customers, Orders, Inventory, OrderItems, and InventoryColors, and Colors.



With mapping out the primary keys, foreign keys, and relationships, we ensured that the backend design supported the actions that we needed. This ERD also helped us to identify any constraints, stored procedures, or updates that we needed to implement throughout development. Along with the ERD, we also created a user flow diagram, which can be seen below.



This diagram helped visualize how different types of users would interact with the system across various pages. The user flow diagram clarified how customers would move from login to product selection, cart, checkout, and order confirmation, while admin and employee users would access inventory management and user administration tools. This diagram became especially valuable as we developed the frontend because it reminded us to implement appropriate redirects, navigation links, and access restrictions for each role. Including both the ERD and the user flow diagram gave us a clear blueprint to follow, helping the development process stay organized and aligned with user needs and system requirements.

We ended up choosing MySQL as our backend platform and outlined table structures for Users, Customers, Orders, Inventory, and Archives. We also planned necessary stored procedures to handle common tasks like creating users, archiving orders, and updating inventory counts. This period also involved setting up our project repository on GitHub and ensuring that we were both aligned on using Git for version control.

Then, we began the coding phase. On the backend, we built the Flask app structure, defined routes and API endpoints, connected to the MySQL database, and integrated stored procedures. We also implemented core logic like secure password hashing and session management to support role-based access and authentication. On the frontend, we developed the HTML, CSS, and JavaScript interfaces for browsing products, customizing orders, managing the cart, placing an order, and handling form validation. We paid special attention to following the wireframe built by the product managers and creating a user-friendly design that has potential to later be expanded.

We then focused on testing and quality assurance (QA). We conducted QA reviews, tested all user flows (including login, product browsing, order submission, and inventory updates), and debugged issues like cart data mismatches, product card loading errors, and price display glitches. Through code reviews and collaboration, we ensured that both frontend and backend components worked together smoothly. Towards the end, we had to prepare our application for deployment. This included refining the startup.py and setup.py scripts to help with database setup and application startup. This also involved writing a detailed README with setup and maintenance instructions as well as compiling all of our documentation and work for final submission.

In addition to these main phases, one important part of our development process was learning to resolve issues and debug as we faced problems with our application and code. There were times when we ran into technical challenges with integrating certain backend features or unexpected bugs that caused delays, but we were able to work to solve them and stay on track with our tasks. For example, when certain data wasn’t displaying correctly on the frontend, we had to revisit both our Flask routes and the database queries to diagnose where the mismatch was happening. These moments of troubleshooting strengthened our collaboration and helped us build stronger debugging skills, reinforcing the importance of flexibility and teamwork in projects like these, preparing us for similar real-world scenarios.

Throughout the project, we also worked with our product manager, James, who played a big role in keeping our team organized and aligned with the overall project timeline. James set up a Jira board to track our tasks, which helped us break down our goals and keep track of what needed to be done. We also recorded the hours we spent on each task to help track our efforts. Regular communication with James and each other, including Zoom meetings during class time, provided us with opportunities to check in on our progress, get feedback, and adjust our work as needed. This collaboration gave us experience working within a structured project management framework and helped us to practice professional communication in a team environment. With the help of James and our consistent communication on the development side, our development roadmap was organized and successful and had a great balance of planning, technical development, debugging, and refinement. With these steps, we were able to meet South Balance’s business goals while considering CMMC requirements and setting the application up for future system changes and growth.

## **Code Repository & Overview**

[**GITHUB**](https://github.com/chase-larocca/CPST-4710-Capstone)

The team has maintained a clean and consistent codebase throughout the project lifecycle. The codebase has been organized to minimize deployment time and make the application as simple as possible for the deployment team at South Balance. When deploying our application, the first destination will be the README (*README.md)* file. This section provides critical insight into many aspects of the application, from initial deployment to maintenance.

After reading this file, the setup file (*setup.py*) is the next stop. Written in Python, this file compiles many individual SQL statements for creating the tables and various stored procedures in the application's backend database. When running this file, the end user will enter some information about their server. Using this information, the script will automatically generate the necessary tables and stored procedures that the website utilizes. The TTOps database is unseen within our repository as this is generated upon initial configuration, but it is vitally important to the application's functionality. A short script written to provide a hash in case the user's table is destroyed is also included in the root directory. Using this file, the systems administrator can quickly generate a hash necessary to create an account in SQL, allowing emergency access manually.

The startup file *(startup.py)* is the center of our repository. This file is the central driver of our Flask server backend and handles most calls to the SQL database, the backbone of systems such as inventory, user creation and editing, login validation, and more. In our case, Flask is primarily used to define routes to different pages and handle passing information from the website properties stored using temporary JSON files to the MySQL database through pre-defined stored procedures within the database mentioned above. If the team could improve this system further, we recommend breaking the single file into separate files, each storing the routes and functions associated with its pages.

Moving on, we will cover the first of three mission-critical folders. This folder, labeled templates within our repository, stores all the HTML files used to build the pages seen when accessing the website. Much of the HTML code is ultimately populated by our JavaScript resources, but our HTML files act as the containers for our scripts to populate. Some hard coded elements, such as the navigation bar, can be seen consistently throughout most files. These HTML files are the reference point for several Flask routes in our startup file.

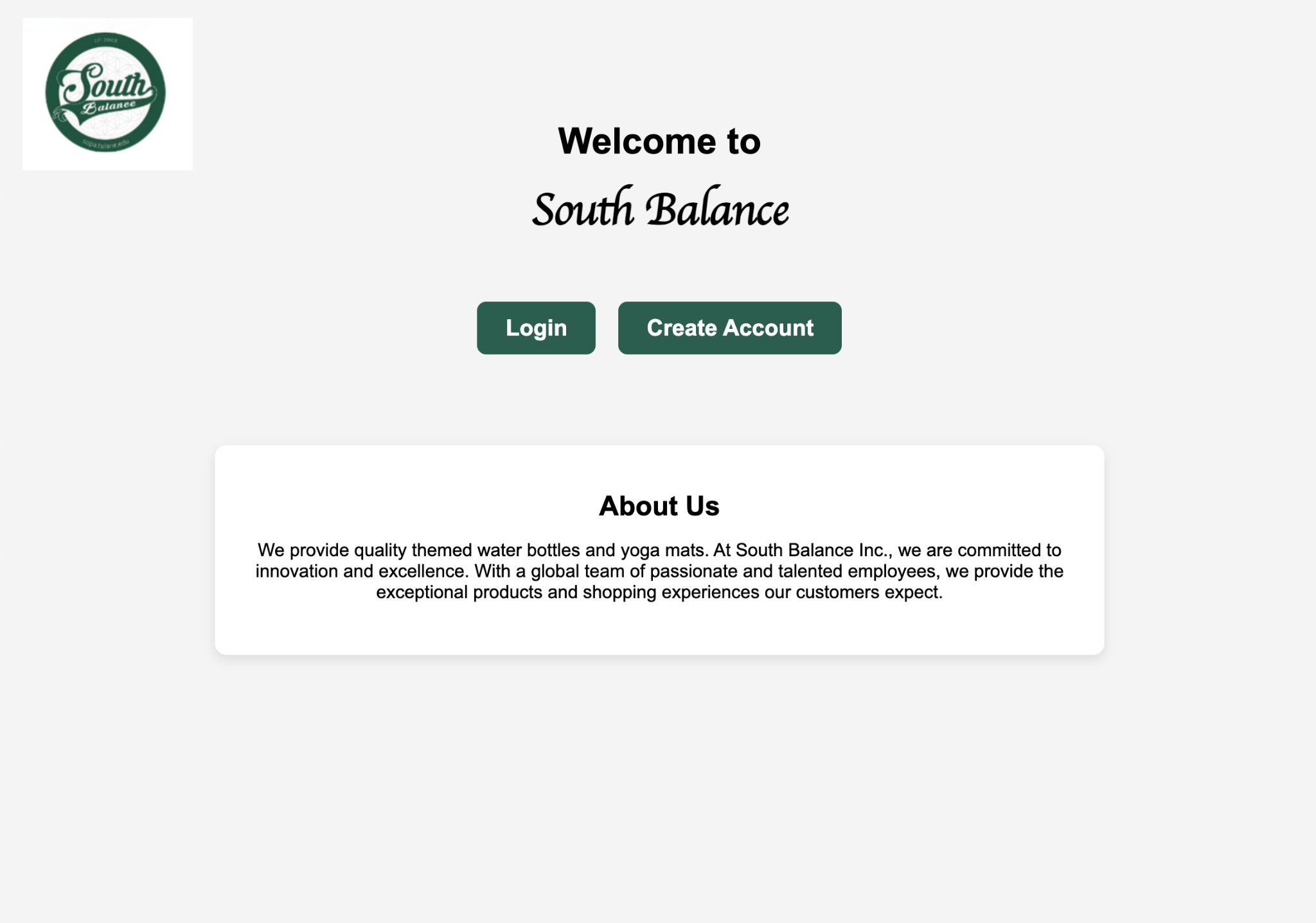
In addition, we have our statics folder. This folder stores the application's JavaScript resources and the CSS files associated with the HTML, and it contains a folder storing the thumbnail reference images. Most pages on our site have an associated style sheet, save a few that require minimal unique styling. The style sheets apply critical design elements that facilitate a streamlined and modern user experience.

Finally, the application's last essential folder is the JavaScript folder (*js).* This folder houses the JavaScript files, which populate much of the HTML by utilizing the data from the application’s Flask routes in the startup file. For reference, the entire products page and the product cards displayed are populated via the associated products file ([*products.js*](http://products.js)*).* This file, in particular, not only builds the HTML code for the product cards but also builds the local storage, which contains data about the items the user is adding to their cart. This information is all pulled and ultimately returned in some form to the database due to the interconnection between the application’s JavaScript and Python-based Flask routes.

This folder further illustrates how critical the startup file is to successfully executing this application. Ensuring that the database credentials are correct and the tables are appropriately named will help ensure the success of the system startup. In the event of system failures, the team has implemented catch statements to ensure that a clean output is provided and some indication of the potential source of the problem is easily accessible for the user.

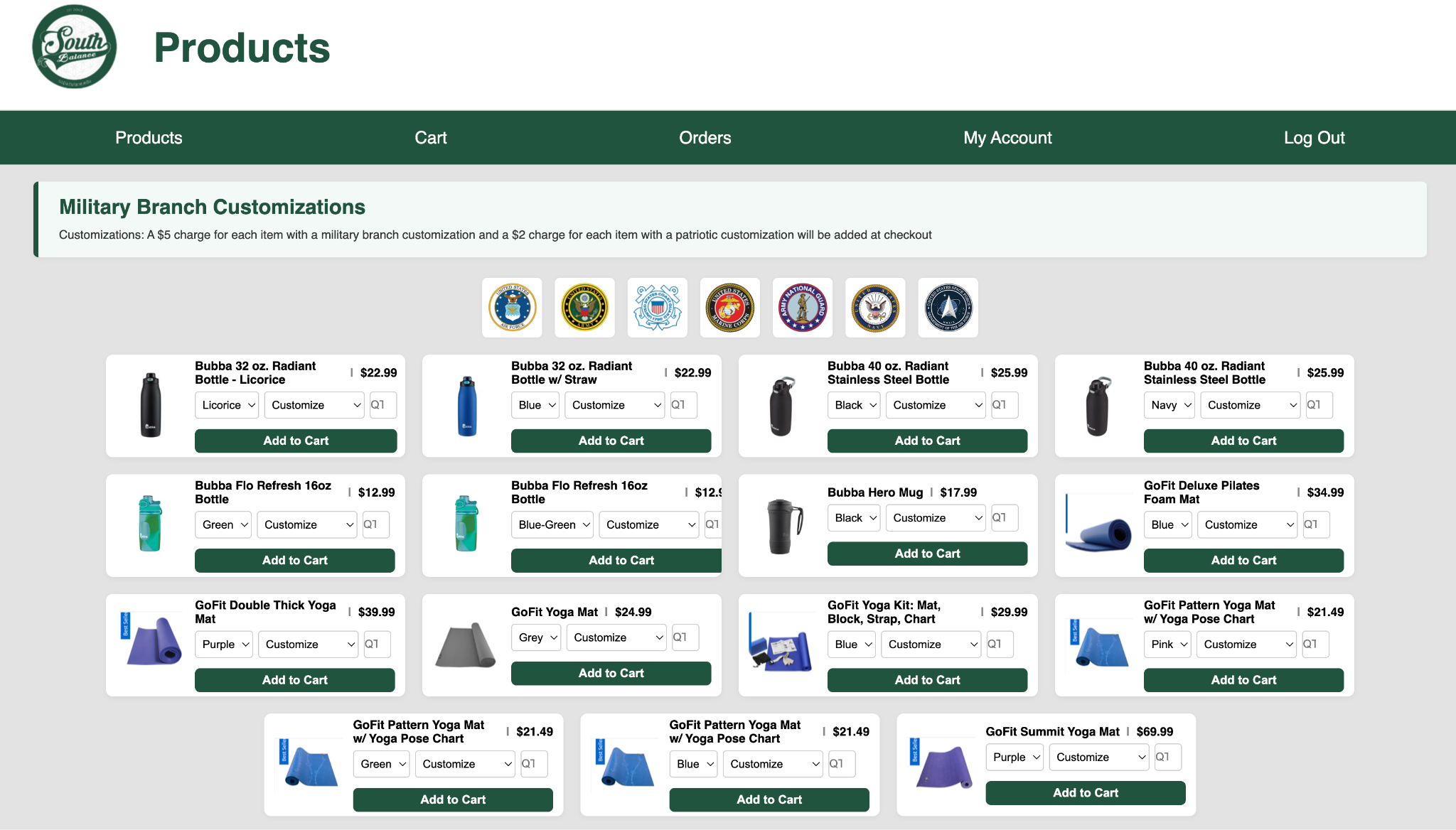
## **Application Walkthrough**

Throughout the development and coding process, we had to ensure that we maintained user friendliness and incorporated all the necessary features within our web pages. We ended up with a homepage, a login page, a page to create an account, a products page, a cart page, an orders page, a page to manage your account, and an inventory page for administrators. The homepage, as seen below, was the first page that we established.

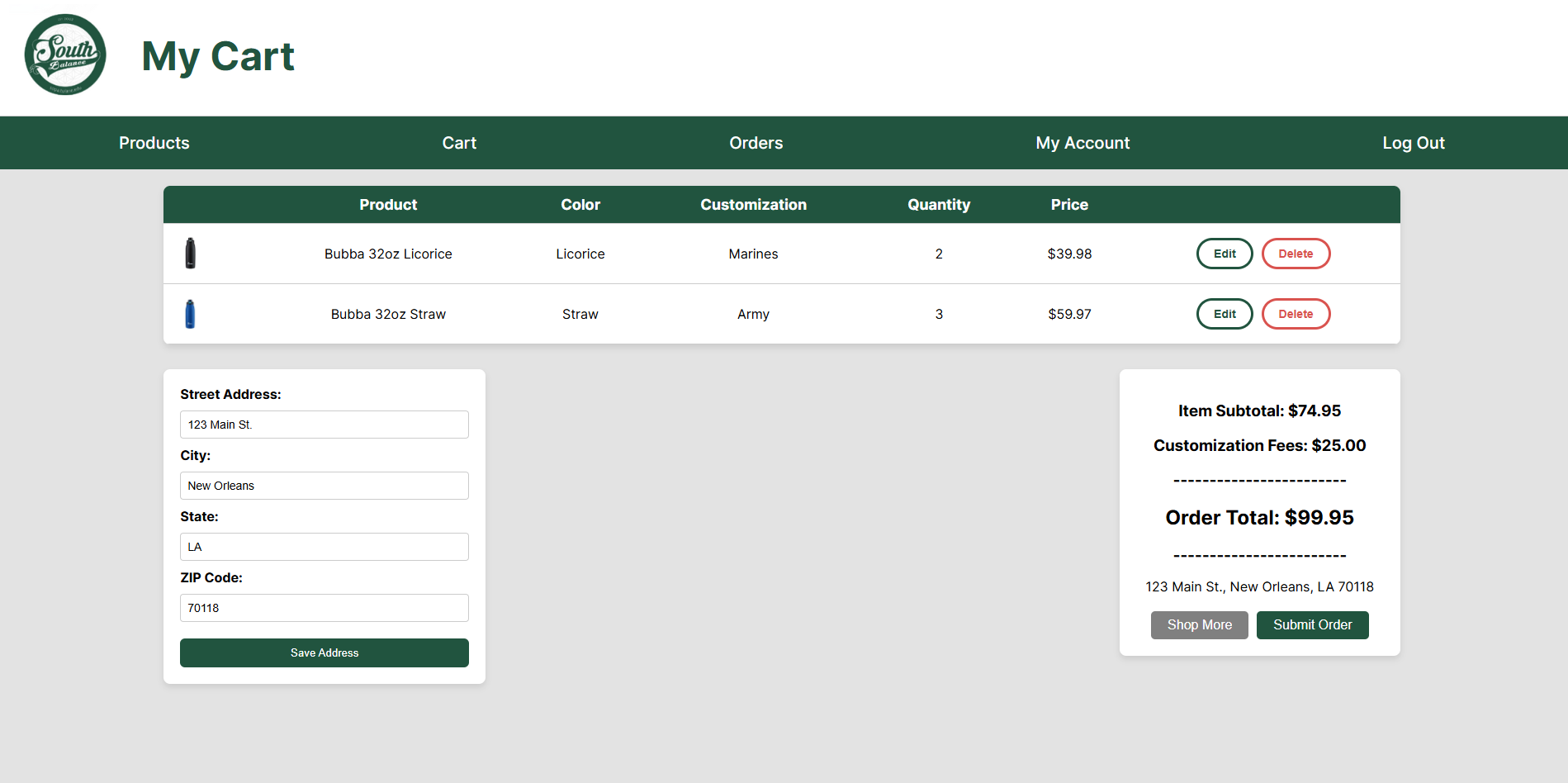


This page serves as the first page users see, welcoming customers and guiding users with what the site offers. This page includes the logo, options to login or create an account, and an “About Us” section that offers a brief introduction to South Balance. This page is important so that users have a good impression of the site and are able to understand how to use it when first opening the application. Here, the users have two options, to create an account, or to login with an existing account. The login page allows registered users to sign into their accounts using their email and password, which are verified against the database. For new users, the create account page makes it easy to sign up by submitting their details, which are then stored securely in the backend system.

Once logged in, customers land on the products page, where they can browse the current inventory, view item details and pricing, and select products to customize and add to their cart. The products page, inserted below, dynamically pulls data from the database, showing real-time availability and ensuring the information presented is up to date.



This product page allows users to browse the available South Balance merchandise, including yoga mats and other items. Each product is displayed with an image, description, price, and available customization options. Once items are added, users can navigate to the cart page, which gives users a summary of the items they have selected, including quantities, prices, and customization details, and offers options to update or remove items before placing an order. This page can be seen below.



After submitting their order, the system processes the order securely, updates inventory levels, and stores order records for future tracking. There’s also the “My Account,” or account management page that allows employees and administrators to view and update account information. Administrators have access to the inventory management page, which provides tools to track and manage stock and oversee operations behind the scenes to ensure everything is working as expected.

## **CMMC Requirements**

A major focus throughout this project was ensuring that the application aligns with the Cybersecurity Maturity Model Certification (CMMC) Level 1 requirements, which are essential for South Balance’s expansion into working with military agencies like the Army and Air Force Exchange Service. Our team prioritized certain CMMC areas because they directly impact the security and trustworthiness of the system, especially when handling things like business transactions and user data.

We focused first on Access Control because controlling who can access which parts of the system is critical for preventing unauthorized changes or exposing sensitive data. By implementing role-based access restrictions (with Customer, Employee, and Admin roles), we ensured that customers only see the shopping and account management pages, while employees and admins have access to more sensitive internal information, like inventory management. This helps limit the risk of accidental or malicious actions, keeping the company’s internal processes and data safe and protected.

Under Identification and Authentication, we implemented a secure login system that uses email and password credentials, also using password hashing to ensure that sensitive user information is not stored in plain text. Secure session tracking maintains user login states and ensures that only authenticated users can access and perform actions within the system. This is crucial because weak or poorly handled authentication processes can easily become areas where data breaches or account hijacking takes place.

For System and Communications Protection, we deliberately maintained a strong separation between the frontend and backend by using Flask’s architecture. Sensitive database logic and operations are kept on the server and never exposed to the client, protecting the system from direct attacks or tampering. This separation helps maintain the integrity and confidentiality of data and ensures that even if the frontend is somehow manipulated, the backend processes remain safe.

Finally, we have not yet fully implemented Audit and Accountability features such as detailed user action logs, but we recognize the importance of this CMMC area for an application like this. Logging and auditing in this case would be very useful for tracking system activity, identifying unusual behavior, and ensuring accountability of someone’s actions, if something goes wrong. Even though these features are planned for future versions,we believe that the system is designed with expandability in mind and allows South Balance to meet even more advanced security requirements as they scale

.

## **Bugs and Resolutions**

Throughout the process, bugs come and go. They are a part of it and can sometimes direct our attention to areas we did not expect. Due to our program's reliance on several systems all working together, there were bound to be bugs everywhere, and as one was resolved, another would appear. For example, upon the initial deployment of the products page, there was a bug that duplicated products if the product had multiple color options. While this manifested on the products page, the issue was tied to the SQL JOIN statement in the startup file. The solution ultimately required separating the SQL statements and getting the available colors into an array with their associated SKUs. Using this, we could populate the color selection dropdown without creating duplicate products.

Additionally, while working on the overall user flow of the system, the team realized that when the user navigates to the “My Account” section of the site, the navigation bar at the top always defaults to the general user’s view. This seems like a simple issue, but it ultimately turned into a complete overhaul of how our system handled its top navigation bar. The previous implementation was rudimentary and involved the same code for the bar on every page. We changed this by turning the navigation bar into a template housed in its file, then using the Jinja2 library, we are dynamically populating that template on our page. By doing this dynamically, we can pass conditions to our pages as they are called. This functionality allowed us to introduce context flags to the links. These context flags would then be passed to the Jinja2 scripts running within the navigation bar’s HTML, where conditions are built to display based on the logged-in user’s role.

Another issue faced included an issue where the login page was authenticating users by email, but the page said “Username,” making it confusing during testing. This was easily changed by updating the label on the HTML page to “Email” to match what the system expected. This ensures that users enter the correct information and are able to successfully access their accounts. Additionally, there were some issues due to unfamiliarity with MySQL and SQL overall. This involved the product cards and some other features not loading properly, which ended up being due to the inventory and colors tables not being loaded properly. After running this properly, the issue was resolved. This highlights the importance of documenting database changes and learning to adapt to new languages and technical skills that the project requires.

## **Post-MVP Recommendations**

The team believes many additional features could be implemented post-MVP to facilitate a more refined user experience, fulfill more CMMC requirements, and enhance security. As the team progressed through the product development process, several features were identified that we would like to recommend be implemented following the deployment of the MVP version of the South Balance website.

Many of the highlighted features recommended for implementation are centered around the application’s user experience. The website’s current form facilitates the core functionality critical to any e-commerce website. That functionality is the ability of a user to go from selecting a product within South Balance’s inventory, providing their shipping information to South Balance, and then submitting that order. While this process occurs, all the data associated with this transaction is retained within the website's backend database. This data has the capacity and flexibility to expand user capability beyond the core functionality the team has already implemented. For example, the team would recommend extending the order history and tracking. This goal would be achieved by providing filtering tools enabling users to search orders by one or more order properties.

The team has also considered moving the completed orders to an order history section, which would help reduce clutter and buildup on a user’s order management page when the system is put into production. Additionally, this expands user capability, allowing users who generally order on a larger scale to manage complex orders with ease. The ultimate goal is to drive returning users to the platform because it is easier for the end user than alternative vendors.

In conjunction with these updates to the visuals and flow of the website. The team recommends implementing additional processes to provide email notifications to customers. These notifications should start when the customer initiates an order and then provide updates until the order is fulfilled. This allows customers further insight and transparency throughout the entire process. This would also eliminate a large number of emails being sent to the order support desk by customers inquiring about order statuses.

Additionally, to further support a streamlined user experience. The team recommends implementing mobile viewing support. Mobile devices are some consumers’ primary method of utilizing e-commerce websites, similar to South Balance’s. Allowing users to shop using mobile devices could drive more user spending and increase South Balance's profit. The mobile version of the site would ultimately flow in a very similar way to the desktop version. However, the current format is very horizontal, so modifications primarily to the product's screen to be more vertical will be required to ensure a mobile-friendly viewing experience. However, all elements are very containerized, so stacking them into a vertical format would not drastically alter the site's appearance.

Furthermore, the team has not neglected to develop potential improvements for South Balance’s backend management section. In the current build of the South Balance website, employees can see all available stock, edit the stock, see current orders, and manipulate current orders. Administrators have the same permissions; however, they can add and edit other users. This fulfills the core functionality of the backend management portion of the application. The team has nonetheless taken note of several features that could improve order tracking, user management, and inventory management. Like the email-based processes outlined in this document, the team recommends email-based alerts for stock changes. Employees should receive stock alerts when inventory is low or reaching the already implemented restock threshold. This allows for more strict management of the products, preventing items from going out of stock.

Regarding inventory and product management, the thumbnails for product images are currently being pulled from a folder storing all of the product images organized by name using the associated item SKU. We recommend implementing a process within the inventory management page, specifically for the edit and add buttons. When generating or editing a product, the employee should be able to upload a new photo, which will be housed in a content delivery service, and the photo should then automatically be given the name of the SKU associated with the manipulated product. Through this process, the current process for associating an image with a product card would not need any significant changes.

Finally, the team's last recommendation concerns user management and expanding our current Role-Based Access Control (RBAC) system. In its current form, the options for user permissions are limited. As new processes and systems are introduced into the ecosystem to maintain the security and retain the CMMC level we have currently achieved, it is essential to consider the implementation of more granular permissions. By this, we mean the ability to limit a user to not just an overarching predefined role, but to allow admins to allocate very specific permissions to individual functions within the website. In conjunction with this, the team believes that implementing log files for tracking user movement and action throughout the site could help identify and patch potential vulnerabilities that arise as the product is tested in a production environment.

## **Reflection & Conclusion**

Reflecting on this project and the process, we encountered several challenges that helped us grow as developers and collaborators. One of the biggest technical challenges we faced was making sure that our frontend and backend components stayed properly connected as we added more complexity to the application. For example, integrating product data between the MySQL database and the Flask routes required careful coordination. We also had to make sure that stored procedures were called correctly from the Python backend, passing the right parameters and handling the returned data smoothly. Debugging mismatches between the frontend display and the backend taught us a lot about the importance of clear API design and consistent testing.

Additionally, we learned a lot from the experience of collaborating through GitHub, managing branches, and reviewing each other’s code. While we worked on separate parts of the system, we constantly had to merge our work and resolve conflicts, which gave us a better understanding of real-world software collaboration. We also deepened our understanding of secure web application development, especially when it came to implementing secure session handling, password hashing, and role-based access controls to meet CMMC requirements.

Overall, this project not only gave us hands-on experience in software development, but it also taught us how to balance functionality, security, and scalability, which are all necessary features and considerations in real-world applications. We are proud of what we accomplished with the MVP version of the South Balance ordering system, and we are confident that the application’s current architecture would be able to support future improvements, including enhanced security measures, expanded features, and more CMMC compliance.