THE AMERICAN UNIVERSITY IN CAIRO

الجامعة الأمريكية بالقاهرة

Software Engineering

Final Milestone

Group 3

Submitted To: Dr. Sherif Aly

Submitted By:

Abdalla Elragal - abdalla.elragal@aucegypt.edu - 900171502

Ali Salem - alisalem@aucegypt.edu - 900183226

Ramy Badras - ramybadras@aucegypt.edu - 900194248

Salma Laban - salmalaban@aucegypt.edu - 900191369

Sherif Wessa - sherifwessa@aucegypt.edu - 900203171

Tayam badawy - tayambadawy@aucegypt.edu - 900183911

Table of Contents

Project Progress Summary

For the duration of this project and throughout the multiple sprints we've gone through, the requirements and goals of this project have become clearer. From the start, the main aim of this project was to cut down on the amount of time AUC students, faculty and staff spend waiting for their food to get ready. We especially found this to be a problem during times with a surge in the number of customers the vendors have to serve such as during assembly hour. Therefore, through this project we hoped we'd be able to give the vendors on the AUC campus a headstart to be able to serve a larger amount of customers without being overwhelmed and cutting down the time the customers have to wait in order to receive their items.

The application can also be used to keep track of previous orders, without having to keep the physical receipt, give feedback and ratings to the vendors according to the quality of their services. Have the items delivered to a specific pickup location on campus for convenience.

Getting started with the development of this app we created a set of user stories in our first sprint to first get a grasp of how an online food ordering system should work, we then created the set of functionalities to satisfy those user stories. For the first sprint we were able to create 7 different user stories, from those we then were able to develop our sprint backlog which included 7 different functional requirements. Later we were required to develop a set of test cases for the features we wanted to implement in our first sprint. In order to get a better visualization of how our functions and processes interact with each other we developed multiple diagrams to help with the visualization. A class diagram was created to get a better understanding of how the different classes of our app will interact with each other and what each

class needs in order to function properly as required. Furthermore, multiple sequence diagrams were developed in order to better visualize the sequence of processes and actions executed by both our app and the customer. Although during the first sprint the diagrams were not of high accuracy, later on during the second sprint we had a better understanding of how they should work and the diagrams were redesigned for the better. Finally, we created a prototype for our app using all of the material we developed during the sprint. Although we were unable to include all the desired features in the prototype, we made sure to include it in later prototypes.

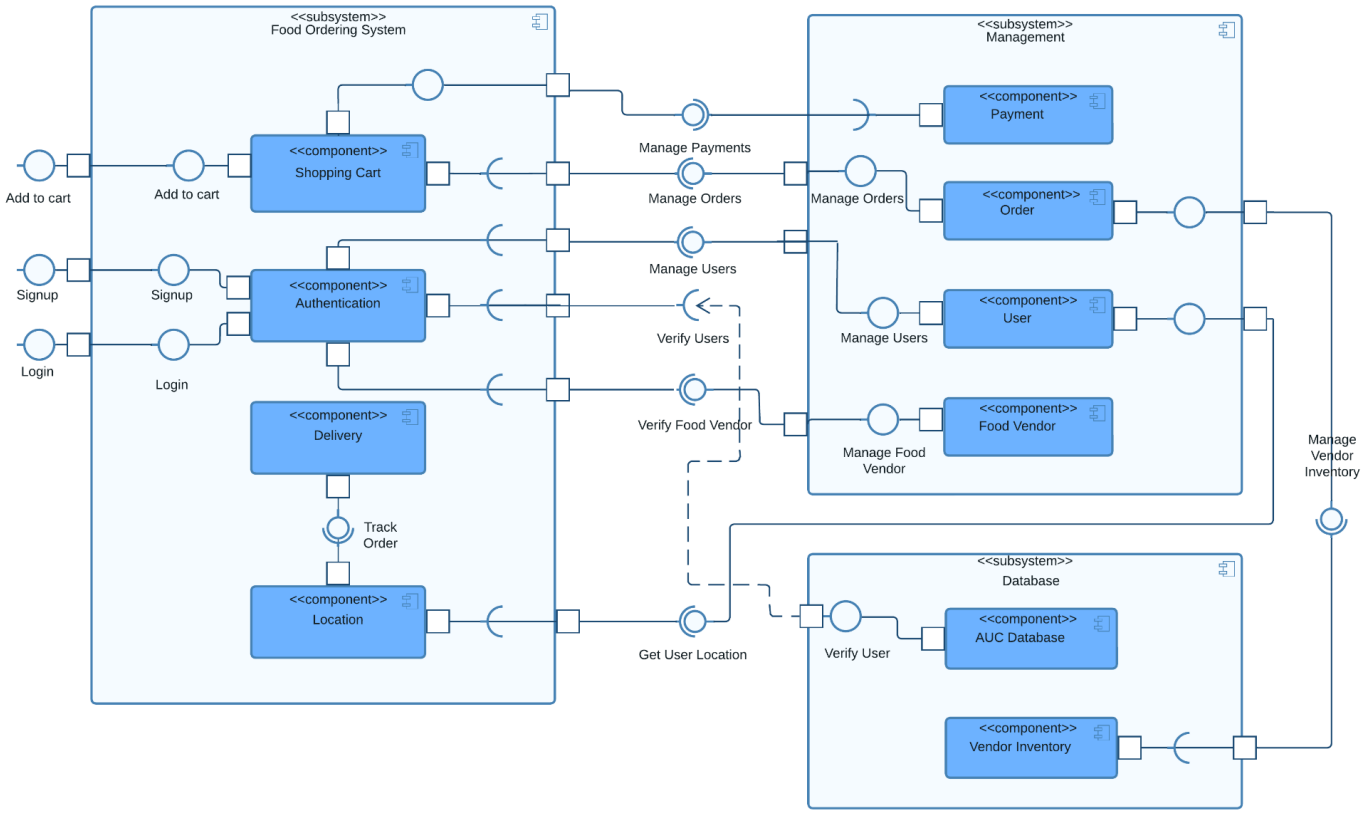
For the second sprint, we had to make sure that by the end, we should have included all of the features, including the ones we couldn't fully cover or prototype in our first sprint. During the second sprint we had to implement 11 different functional requirements and similarly to the first sprint, we later developed user stories, test cases, component diagrams and sequence diagrams for said features. This time the user stories we improved and the features were each given weights as to how much time / resources it would take to develop them. Similarly the test cases were improved as we were able to include more test cases for each feature of the app. The class and sequence diagrams were both improved and included more information than the previous ones using our latest knowledge. Finally we developed another prototype for the app which includes all the features we wanted to implement using a different prototyping tool we found to be more suitable for our requirements. Finally we roughly calculated the team's velocity in order to be able to have an overview of how much work the team might be able to get done by the end of the sprint. With better time management and work distribution during this sprint we were able to achieve 100% of our sprint goals.

Finally for this milestone we included the final and more detailed diagrams that best describes how our app should function, including the details of the software architecture we

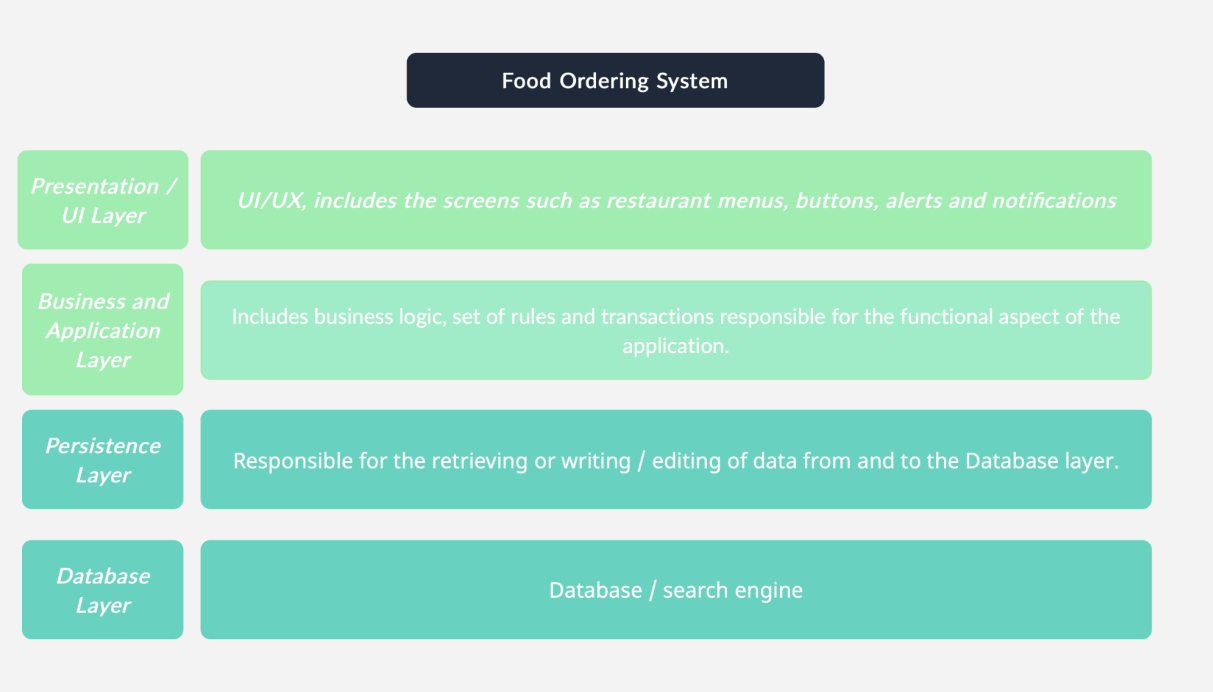
found most suitable for the app. The diagrams include component, sequence, class and use case diagrams. Lastly we included our final prototype. The following were developed with more ease using the materials we had from out previous sprints.

Detailed Architecture

Component Diagram



Layered Architecture of the Food Ordering System



System Design

A layered architecture pattern is most suitable for our application as we have separate layers each with a specific set of rules, dependencies and operations. It's clear that the set of functionalities can be separated and grouped together in different layers. Later the different layers will communicate among each other through specific functions, such as the user clicking on a button that is displayed using the presentation layer, the Business and Application layer then receives the request, processing it and moving it on to the next layer and so on. One of the main advantages of this architectural pattern is that each layer is only responsible for executing and carrying out its rules without knowing how the other layers will work (Richards).

Presentation Layer

The presentation layer is responsible for displaying the user interface for users of the app. Everything visible on the screen including buttons, clickable icons and notifications are in this

layer. If a user clicks on any of the clickable elements of this layer a request will then be sent from this layer to the next layer Business and Application layer. The request goes through all the layers in order to respond to that request.

Business and Application Layer

The main responsibility of this layer is to execute certain business rules that are specific for the request sent from the presentation layer. It receives the data that is required for the request from the persistence layer, then it carries out the appropriate actions, and sends the data back to the presentation layer (Richards). In other words, this layer includes all the calculations and the processing that takes place from the received requests from the presentation layer. For example if the user clicks on adding a specific item to their order, the request is sent from the presentation layer to this layer in order to calculate the grand total of their order and sends it back to be displayed in the presentation layer.

Persistence Layer

This layer is responsible for accessing and manipulating the database to get the required data and send it back to the business layer. This layer contains all the codes and algorithms that allow it to retrieve all the data that are needed for the request (Morlion, 2018).

Database Layer

This layer includes all the data the application requires to function, such as the AUC email database to verify that the user is using a valid AUC email address. The database includes the menus, prices, names, previous or ongoing orders etc. Data is continuously fetched or edited from this layer for example to get the price of an item the user added to their cart, or update the price of an item.

Physical Architecture

The physical architecture is the arrangement of physical objects and interfaces to satisfy the logical requirements. Therefore, our system will require some physical elements such as:

- Servers: to store system database and users' data
- Hardware: For using the application as well as entering and updating data
- Internet/wifi connection and routers: for communication between user and system.
- Navigation system: to identify the location of the user/food station to deliver food.

Detailed Design

Functional Requirements

1. **Sign up**

   The user must sign up with the AUC email to ensure that the user belongs to the AUC community (staff, faculty, students) and a password containing 8 characters with 1 uppercase letter and 1 special character.

2. **Logging in**

   The user uses the email and password that was used when signing up.

3. **Verify password**

4. **Navigate menu**

   A menu used to make the application more user friendly that the user can use to maneuver easily through the application.

5. **Select item (Add to Cart)**

   The user must select 1 item at a time and add it to the cart.

6. **Customize item**

   After selecting an item the user has the ability to customize what they ordered (if applicable) for example a sandwich without olives.

7. **Remove item (Remove from Cart)**

   An option if the user wants to remove an item from the cart the user must select 1 item at a time or delete the whole cart.

8. **Selecting location**

After the user has confirmed his order, he should choose whether to pick up the order from the food vendor or the designated pick-up stations at each building. If the user chooses the pick-up stations, he has to also select the building.

9. **Pay for order (Credit or Debit Card / Auc Pay)**

   The next step is for the user to pay for the order. The user can do so either by paying by credit or debit card or Auc pay. The user must input the credit/debit card number, name, expiration date and CVV or the user can scan the card and he won't need to fill in this information. However, if the user chooses to pay through Auc pay, he will be directed to AUC's website to pay.

10. **Payment Confirmation**

    After the amount has been paid a message appears on the screen: payment of: X amount has been successfully made.

11. **Order confirmation**

    An email will be sent to the user's AUC email containing the invoice of the order made which includes the order number, the amount paid and the items that have been ordered.

12. **Cancel order**

    The user can cancel an order if the order has not yet been made by the food vendor.

13. **Track the order**

    The user has the option to track the order and know whether the order is still in the food vendor or is on its way.

14. **Pick Up Confirmation**

    After the food has been picked up an email will be sent to the vendor and the customer confirming that the order has been picked up by the customer.

15. **Review Order**

    The user has the option to review the order by giving it a rating on a scale from 1 to 5

16. **Support**.

    If the user faces any problems the user can chat with an employee to help him through the application.

17. **Orders history**

    The user can view the orders history up to 10 orders. The order history contains which food vendor the user chose, the items, and the amount paid.

18. **Edit menu**

    The vendors would be able to add new items or remove old ones as well as update the items according to inventory (stock).

Nonfunctional Requirements

1. **Reliability**

- The application should handle any number of users in order not to crash.
- The application should notify the vendors once an order has been placed by a customer

2. **Usability**

- UI/UX of the system should be simple and user-friendly to ensure the easiness of the purchasing process.
- The design of the user interface would be consistent with the AUC theme.

3. **Responsiveness / Speed**

- The system should always respond quickly to users' needs, and it should also be flexible to allow any modifications.

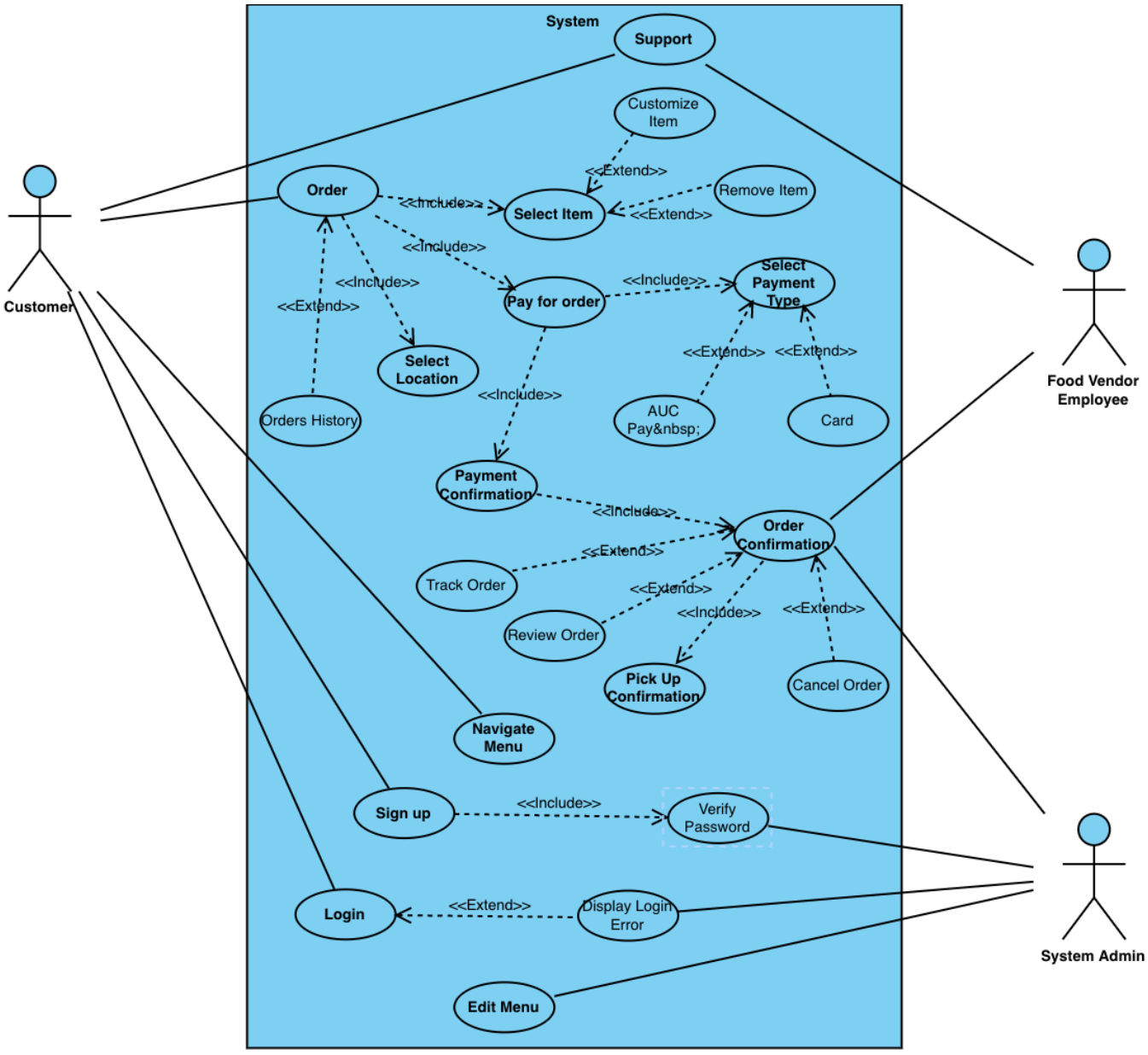
4. Maintainability

- The system should be maintainable to ensure its effectiveness through continuous updates, restoration, and backup in case of any failures.

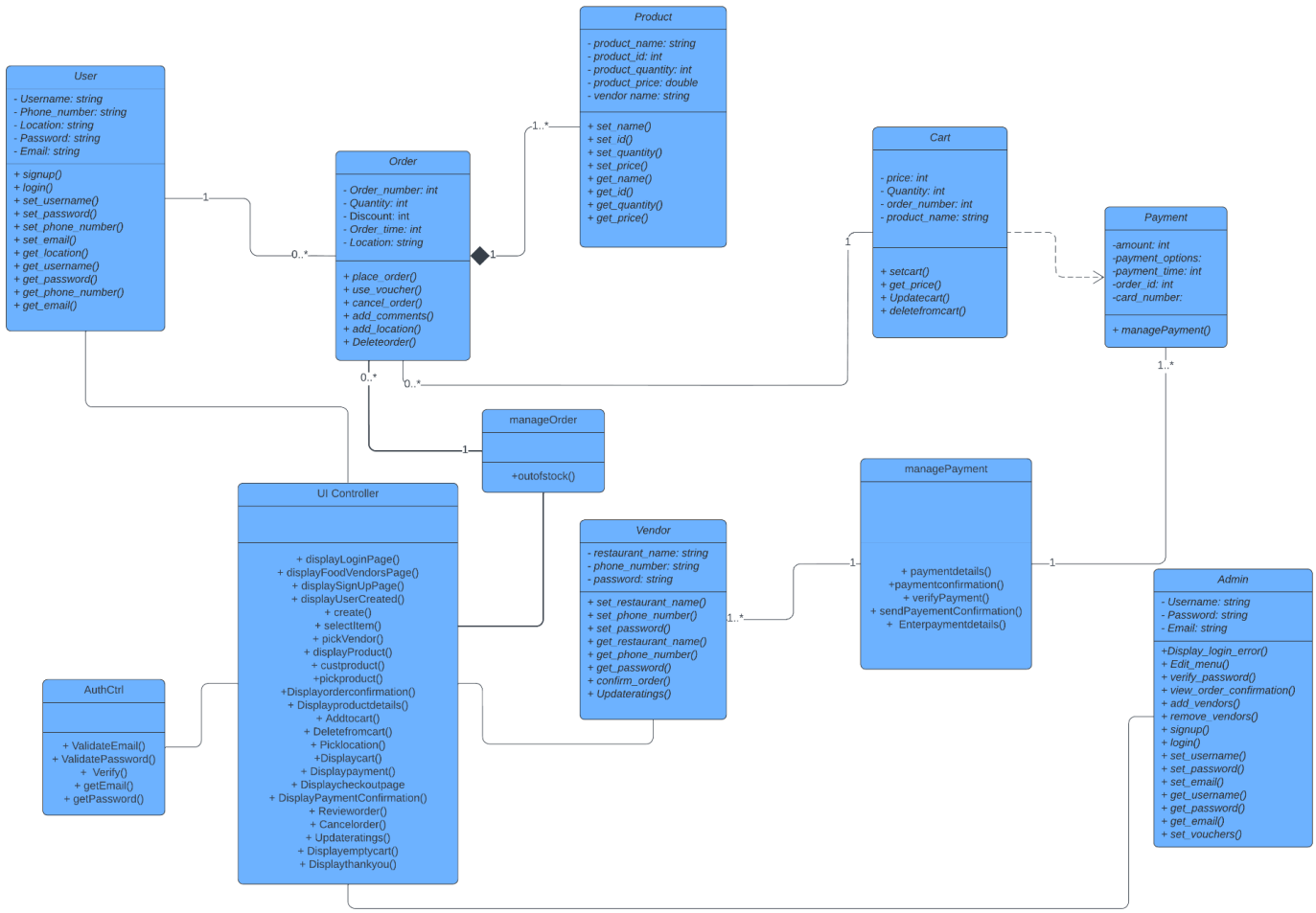
5. Security

- Database firewalls can be used to protect against security attacks
- Customers' information and payment data should be encrypted to maintain a securable system

Set of Use Cases

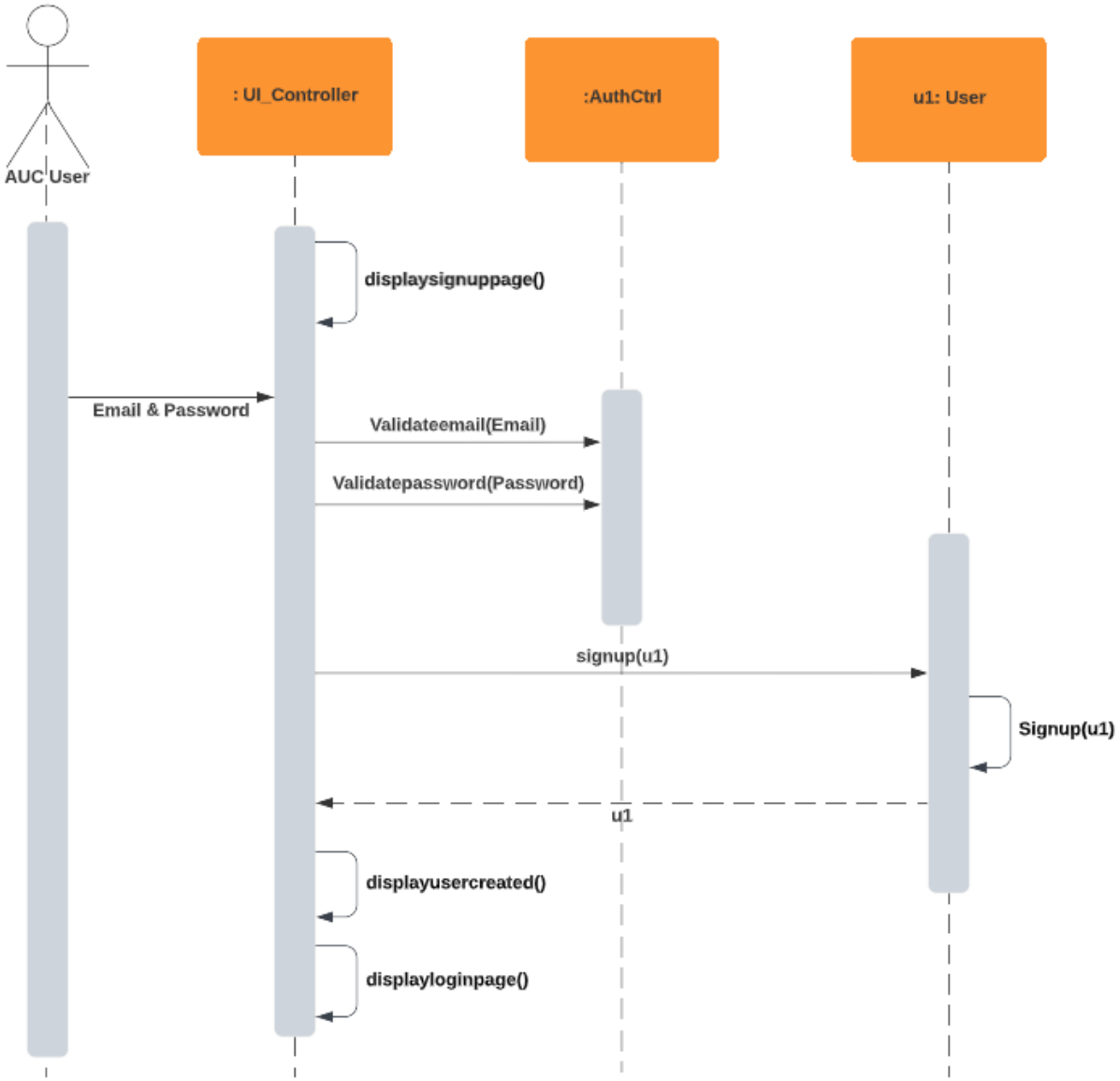


Class Diagram

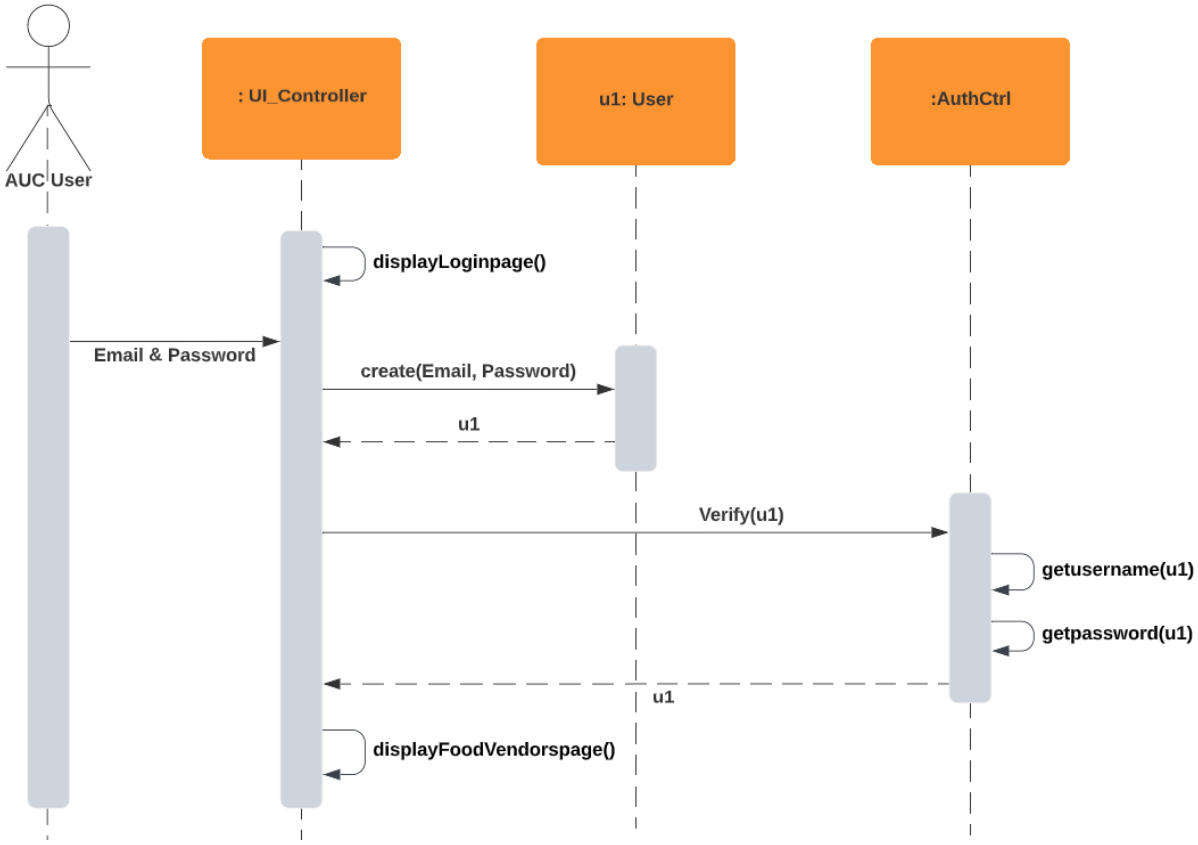


Sequence Diagrams

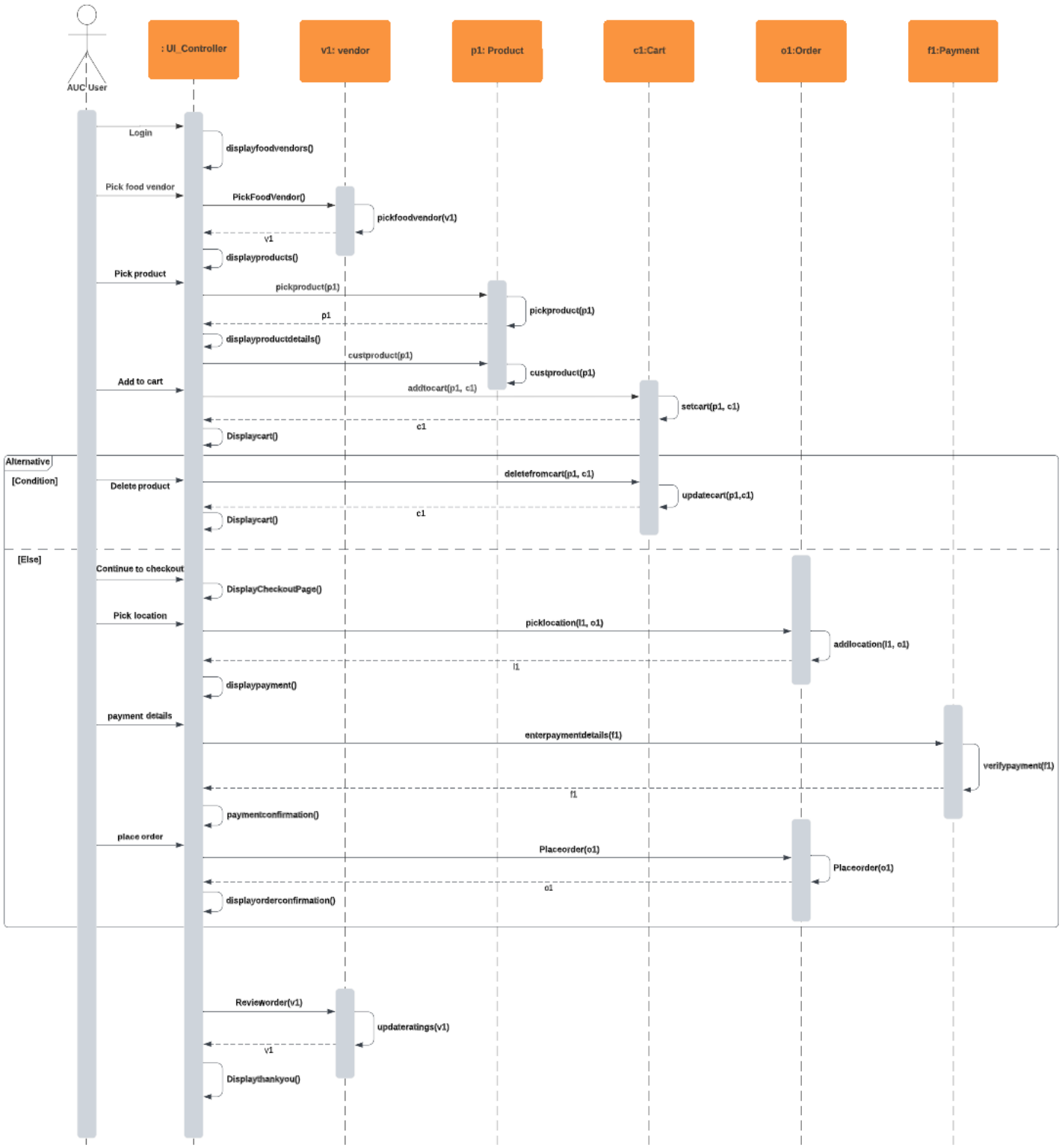
Signup sequence diagram



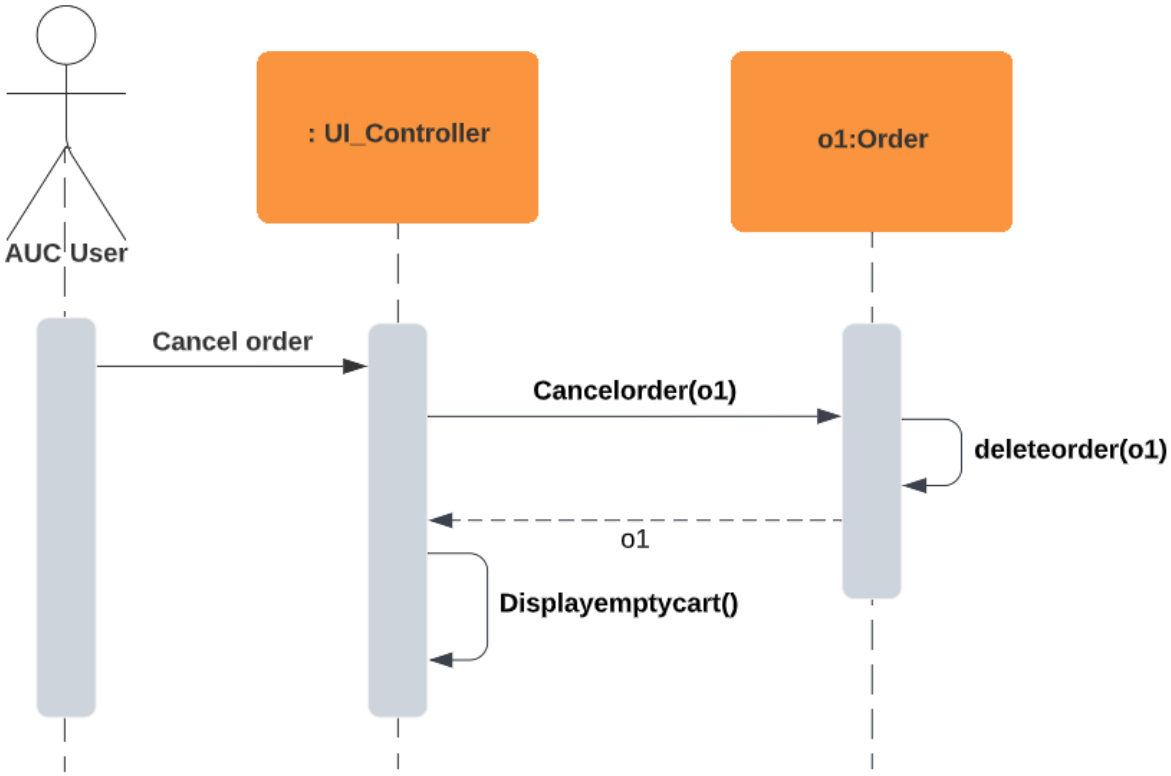
Login sequence diagram



Ordering sequence diagram



Cancel Order sequence diagram



The Refactored Source Code (Prototype)

Figma:

https://www.figma.com/file/5DPIZTZWzw04qiHXLhUtqR/SWE---Project?node-id=12%3A425&t=shZyUijzvs00WOIn-1

Exported files:

https://drive.google.com/file/d/1nLsG441HWL33uR5Tsn1QOzIfvyExg0xv/view?usp=share_link

Plans for Future Expansion

Features to include later on

We have a few ideas of features that may be included in the future for further development of the app and for maximum convenience.

- **Wallet:** each user will have their respective wallet that can be charged using the same methods of payment we currently have supported in the app. The wallet will facilitate refunds and payments as transactions can be done more quickly without requiring credit card information each time.
- **featured page:** the featured page will include any current offers vendors may have, any new item that has been recently added to any of the vendors' menus or any specific items that might be of interest.
- **Search engine:** The plan is to implement a search bar at the top of the home page where the user can insert the name or description of an item and the search engine will return the list of items that fit the criteria and their respective vendors.
- **Quick order:** This feature allows the user to save a preset order of their choice which they can later re-order at any time.
- **Voucher Redeeming:** Allows the user to redeem vouchers for either percentage off of orders from a certain vendor, an amount to be added to their balance, etc…
- **Planned order:** This feature allows the user to set a specified time of day of their choice of which they should have their order ready by that time.

Github Link:

https://github.com/sherifwessa/SWE-Project-Group-3

References

Richards, M. (n.d.). *Software architecture patterns*. O'Reilly Online Learning. Retrieved December 2, 2022, from https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

Morlion ·, P. (2018, June 30). *Software architecture: The 5 patterns you need to know* . dzone.com. Retrieved December 2, 2022, from https://dzone.com/articles/software-architecture-the-5-patterns-you-need-to-k