**Class Project Phase IV**

**CSE 360 - Fall 2022**

Team Number: 47

Team Members:

Jason Roberts

James Weber

Garret Goodwin

Ryan Radtke

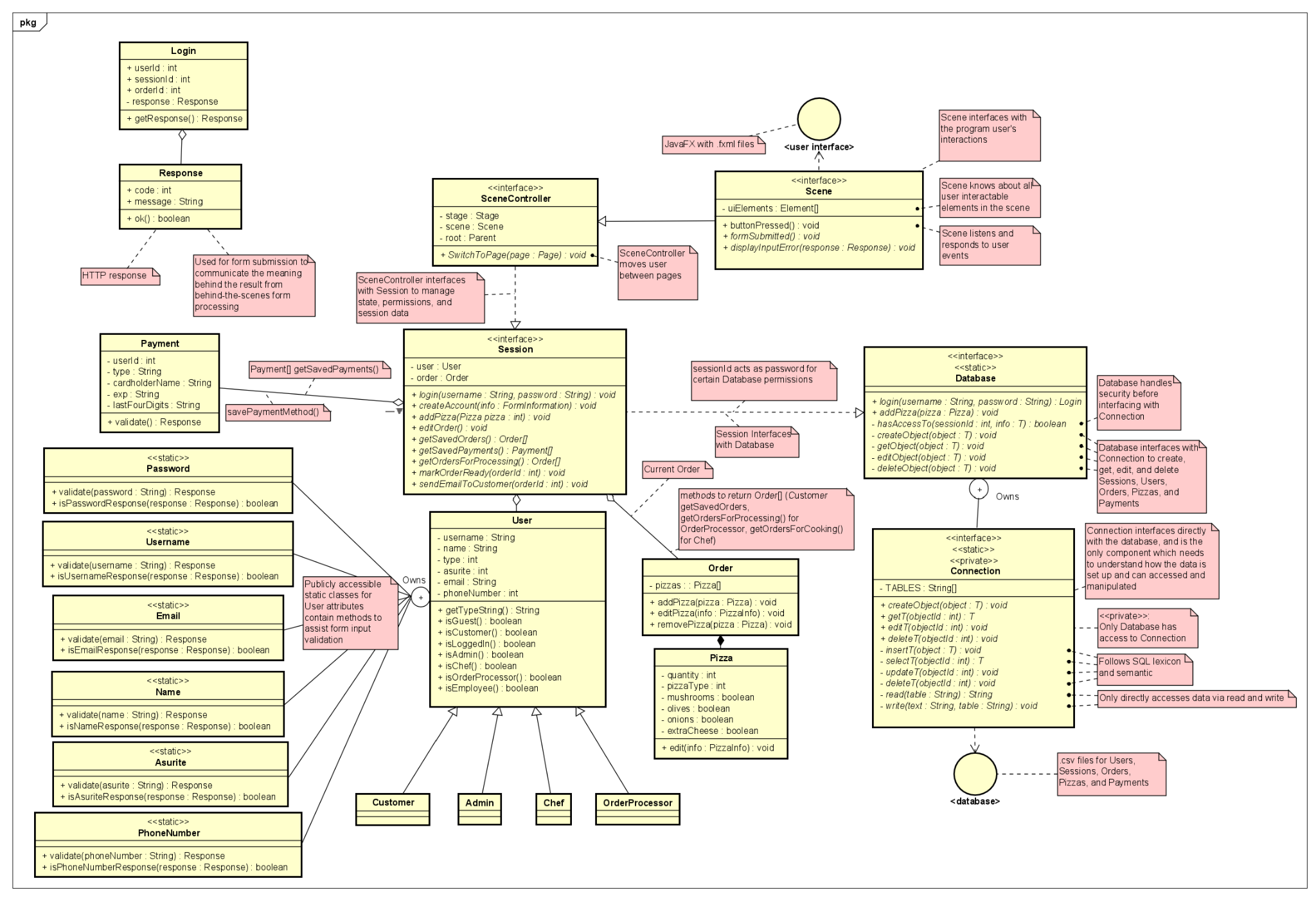
Johnny Duong

1. Link for the technical presentation (around 15 minutes). You can record this as a YouTube video or upload your video to the google drive of your team leader or one of your team members and provide a link to the google drive video

Google drive or YouTube link here:

<https://youtu.be/xUWg8jEPtGc>

1. Final class diagram (copy and paste the Astah class diagram here).

“Owns” relationship describes a nested class, as defined in IBM Rational Software Architect Standard Edition, Version 7.5.5.5

1. Testing report. Use the following format

| Test case | Testing Scenario from Phase II | System Output | Test case passed (Yes/No) |
| --- | --- | --- | --- |
| 1 | Test account creation, make sure all accounts created are validated and any errors are appropriately handled (make input format agnostic and test defensive coding) | Correct Process:    Taken to Pizza Selection Page after successful sign in    Handled Errors: | Yes |
| 2 | Test correct account creation, ensure accounts are properly added to the database |  | Yes |
| 3 | Test account information security, functions have appropriate privacy/accessor settings | All database access must go through the Database class by design. The Database class’s public interface makes the appropriate checks to make sure the user should have access to the requested information, usually via the User’s current Session’s sessionId.    Example: | Yes |
| 4 | Test order handling from multiple accounts, warning message should pop up alerting to any changes that have occurred within the last minute from a different account | We did not implement functionality to handle multiple users manipulating data at the same time. Going forward, this could be implemented by raising an event after data is manipulated, and necessary ends listening for that event. | No |
| 5 | Test order cancellation, ensure order is properly removed from the database | Processor canceling customer’s order    6 indicates status constant Order.CANCELLED    Customer can see their order status is canceled | Yes |
| 6 | Test payment authentication | Correct Process:  Correct Asurite  Correct Card  Card gets added to added to database  Handled Errors:  Invalid Asurite    Invalid card | Yes |
| 7 | Test Order completion, customer should be alerted, and order should go into archive | Order Processor can mark order complete  5 indicates status constant Order.COMPLETE    Customer can see their order status is complete | Yes |
| 8 | Test account login, incorrect and correct information should deliver the user to their respective page | Correct Process:  Login as customer    Customers get taken to the pizza selection page  Login as order processorOrder Processors get taken to the order processing pageLogin as chefChefs get taken to the chef page  Login as adminAdmin get taken to the admin pageIncorrect Process: | Yes |
| 9 | Test order creation, adding pizza options should update price properly and reflect in the final order |  | Yes |
| 10 | Test administrator account authority, administrator should be able to adjust other accounts and review orders and query the database | Yes (Success):  Admin can create users and give them customer, chef, order processor, or admin permissions  Creating new users:      1,2,3,4 respectively indicate user\_type constants Customer.TYPE, OrderProcessor.TYPE, Chef.TYPE, Admin.TYPE  Changing permissions for existing users:    Before:  4 indicates user\_type constant Customer.TYPE    After:  2 indicates user\_type constant Chef.TYPE  No (Initial requirement not met):  Design changes mean admin should no longer need to query the database. | Yes/No |
| 11 | Test database limitations, make sure the database can handle high amounts of users | The way the database is set up as parsing .csv files, growth scales poorly. The system was designed so that only the implementation of the Connection class would need to change when implementing a server and more practical database. Connection’s interface could remain the same. | No |

1. Conclusions (No more than ½ page, single space 12 font size)

The key thing that worked well for our group through the project was communication. We kept each other updated on what we were working on and made sure that each aspect of each phase of the project was being completed. We also were not afraid to reach out to one another with questions or suggestions when we spotted any issues.

One of the major issues we encountered when designing and executing our code for this project was difference in skill levels when it came to coding in the required language and library (Java/JavaFX), but this was quickly overcome with each team member helping each other. Another major issue was the lack of explicitness of the project requirements and lack of feedback, a lot of our design choices were based on other applications that are currently on the market today. Given the circumstances I think this was the best outcome, I do think this project would have benefitted from feedback at each phase so we could better tailor our project to evolving design specifications as necessary.

I would recommend to future CSE360 groups to be much more aggressive towards the project specifications and ask more questions that lead to explicit answers. I would also recommend that the team look ahead to future phases of the project and their requirements so that they can plan accordingly. Certain things like “statistics” were not mentioned as being required until this phase.

As for this project's statistics, our project came in about 5000 lines of java code and 1300 lines of FXML code. Estimated about 90-120 lines of code each day per person on the team and took us about 2 weeks to complete. This does not include the refactor we had to accomplish in the middle of phase 3 to accommodate the data structure we used to save client information.

1. Data Design

Identified data entities

Session

| id | time\_created | time\_updated | user\_id | order\_id | is\_closed |
| --- | --- | --- | --- | --- | --- |
| int | Timestamp | Timestamp | int | int | bool |

User

| id | time\_ created | time\_ updated | username | name | user \_type | encrypted \_password | asurite | email | phone\_number | session\_id |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| int | Time stamp | Time stamp | String | String | int | String | int | String | String | int |

Order

| id | time\_created | time\_updated | user\_id | status | is\_saved | delivery\_method |
| --- | --- | --- | --- | --- | --- | --- |
| int | Timestamp | Timestamp | int | int | bool | int |

Pizza

| id | time\_ created | time\_ updated | order\_id | pizza\_type | mushrooms | olives | onions | extra\_ cheese | quantity |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| int | Timestamp | Timestamp | int | int | bool | bool | bool | bool | int |

Payment

| id | time\_created | time\_updated | user\_id | card\_number | exp | cardholder\_name | cvv |
| --- | --- | --- | --- | --- | --- | --- | --- |
| int | Timestamp | Timestamp | int | int | int | String | int |

E-R diagram

