SEG 2105 - Introduction to Software Engineering
Fall 2022
Android Project: Mealer App

Munir Alsafi (300013845)
Peter Saroufim (300015864)
Rakshita Mathur (300215340)

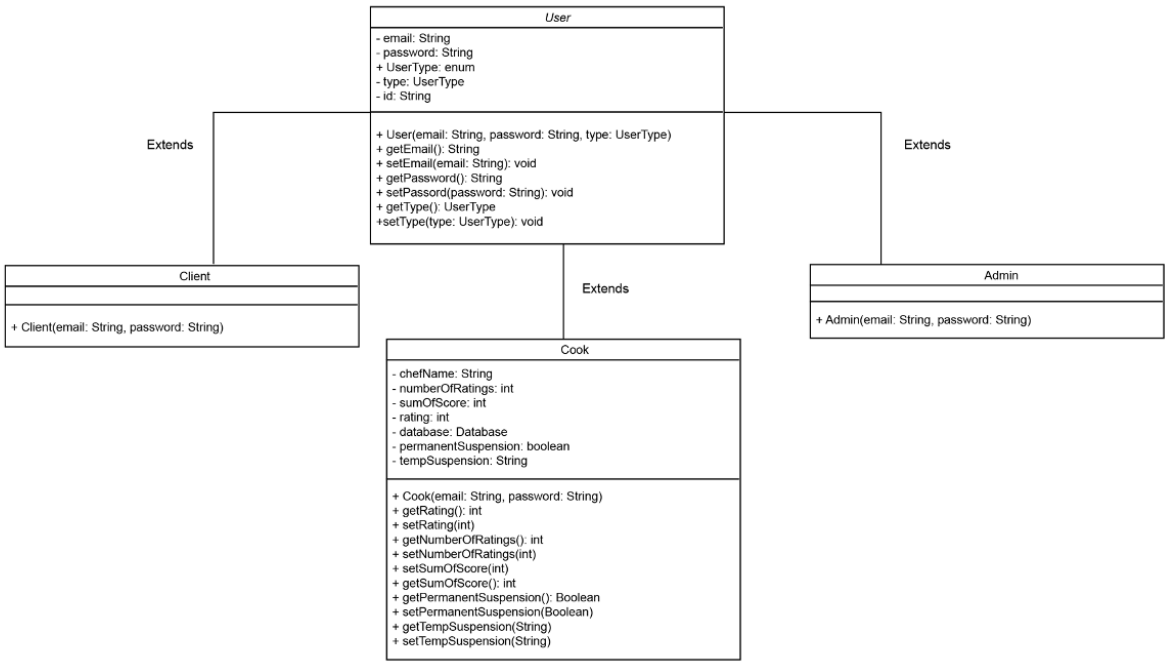
Introduction

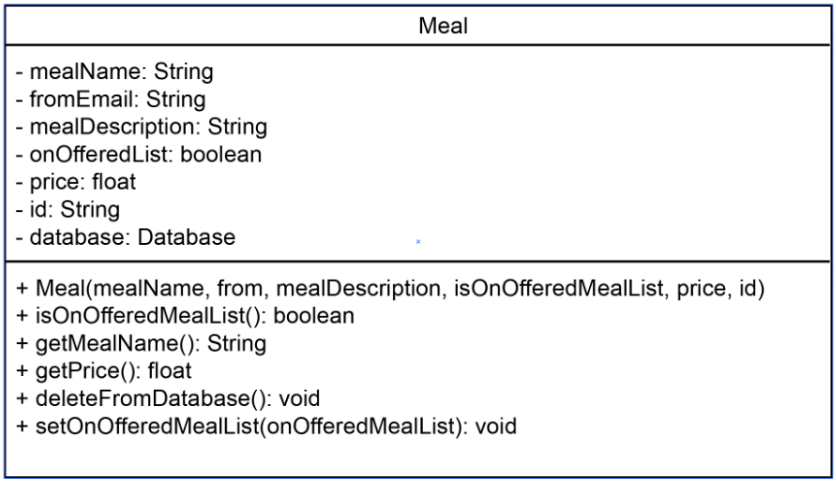
The ‘Mealer App’ is an Android App that connects cooks to clients. Cooks are able to create meals and publish them for users to view. Users are able to view the various meals offered by cooks, place an order, and give the cook a review once their order is approved. If an order was not up to standards, users can submit a complaint to the cook, where an admin will then dictate whether the cook should be temporarily suspended, permanently suspended, or dismissed.

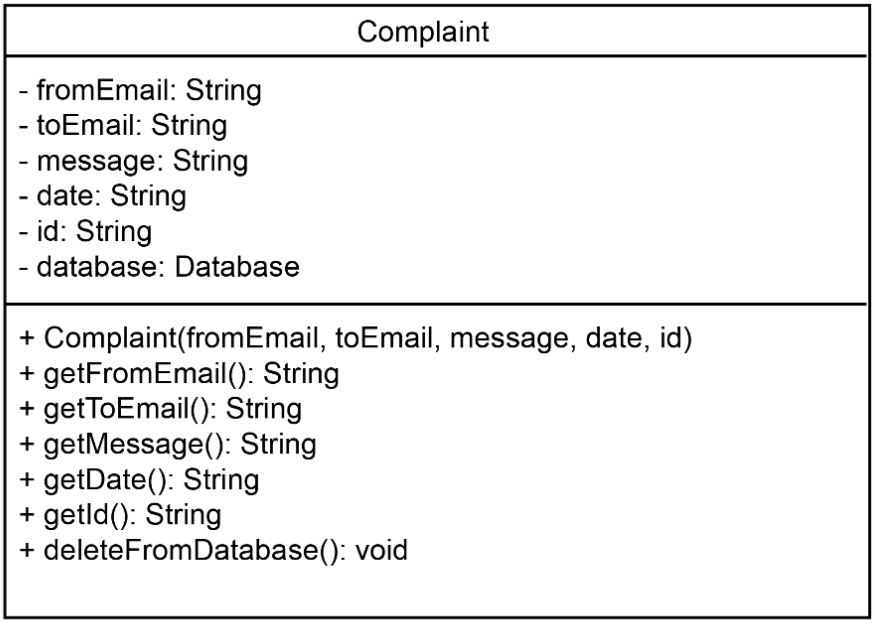
The App was created by three individuals using Android Studio and the Firebase Database application.

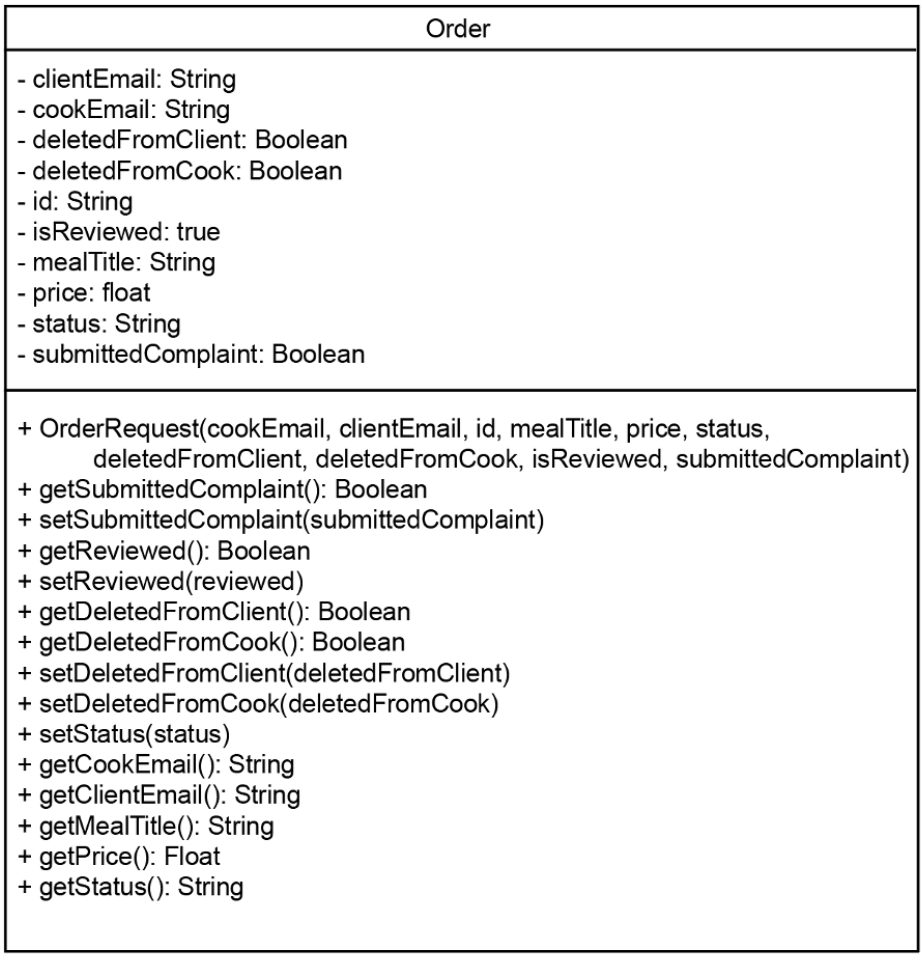
Updated UML Class Diagram

The following are the UML Class Diagrams that describe the objects used in this app.




Meal
- mealName: String
- fromEmail: String
- mealDescription: String
- onOfferedList: boolean
- price: float
- id: String
- database: Database
+ Meal(mealName, from, mealDescription, isOnOfferedMealList, price, id)
+ isOnOfferedMealList(): boolean
+ getMealName(): String
+ getPrice(): float
+ deleteFromDatabase(): void
+ setOnOfferedMealList(onOfferedMealList): void


Complaint
- fromEmail: String
- toEmail: String
- message: String
- date: String
- id: String
- database: Database
+ Complaint(fromEmail, toEmail, message, date, id)
+ getFromEmail(): String
+ getToEmail(): String
+ getMessage(): String
+ getDate(): String
+ getId(): String
+ deleteFromDatabase(): void


Order
- clientEmail: String
- cookEmail: String
- deletedFromClient: Boolean
- deletedFromCook: Boolean
- id: String
- isReviewed: true
- mealTitle: String
- price: float
- status: String
- submittedComplaint: Boolean
+ OrderRequest(cookEmail, clientEmail, id, mealTitle, price, status,
deletedFromClient, deletedFromCook, isReviewed, submittedComplaint)
+ getSubmittedComplaint(): Boolean
+ setSubmittedComplaint(submittedComplaint)
+ getReviewed(): Boolean
+ setReviewed(reviewed)
+ getDeletedFromClient(): Boolean
+ getDeletedFromCook(): Boolean
+ setDeletedFromClient(deletedFromClient)
+ setDeletedFromCook(deletedFromCook)
+ setStatus(status)
+ getCookEmail(): String
+ getClientEmail(): String
+ getMealTitle(): String
+ getPrice(): Float
+ getStatus(): String

Contributions of team members

The majority of the project was completed synchronously together. We would meet together in person and complete the project on a single laptop. Since there was a large variation in experience, it was determined that the team would learn the most if tasks were not split.

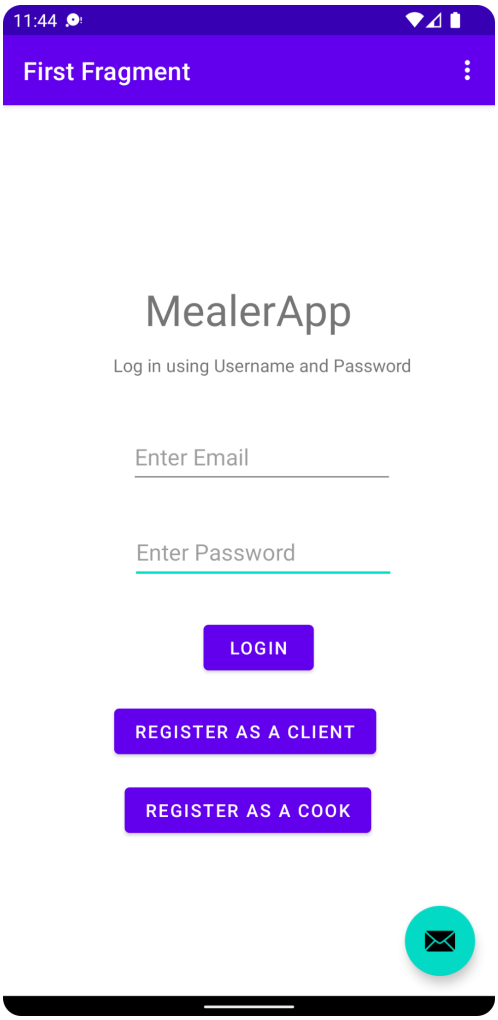
Deliverable 1	
UML Class Diagram	Munir
User can create a Client or Cook account	Peter
Administrator, cook, or client can see the “welcome screen” and their role	Rakshita
User can log off	Munir
Field Validation	Rakshita

Deliverable 2	
UML Class Diagram	Peter
Storing account information & complaints in the DB	Munir
Administrator can view list of complaints	Peter
Administrator can action complaint	Rakshita
Cook can view message informing of their suspension	Rakshita

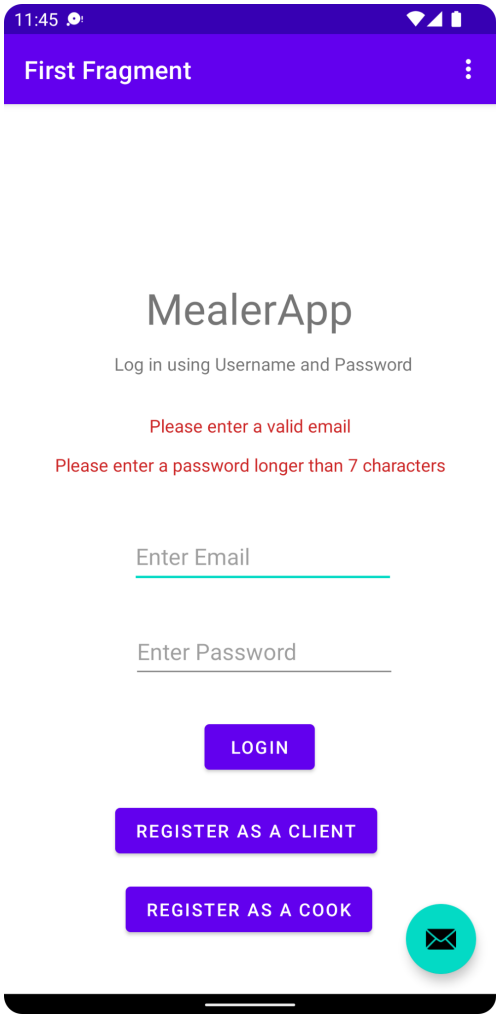
Deliverable 3	
Updated UML Diagram	Munir
Cook can add a meal to the menu	Munir
Cook can add meal to offered meals list	Rakshita
Cook can delete meal from the menu	Peter
Cook can remove meal from offered meals list	Peter
Cook cannot perform any action when suspended	Rakshita

Deliverable 4	
Updated UML Diagram	Munir
Final Report	Rakshita
Client can search for meal by non-suspended cooks	Munir
Client can view Cook’s information and rating	Munir
Client can submit purchase request	Peter
Cook can receive purchase request	Rakshita
Client can view status of their purchase	Munir
Client can rate the cook when they purchase a meal	Peter
Client can submit a complaint to administrator	Peter
Cook can approve/reject purchase requests	Rakshita
Cook can view their profile and rating	Rakshita

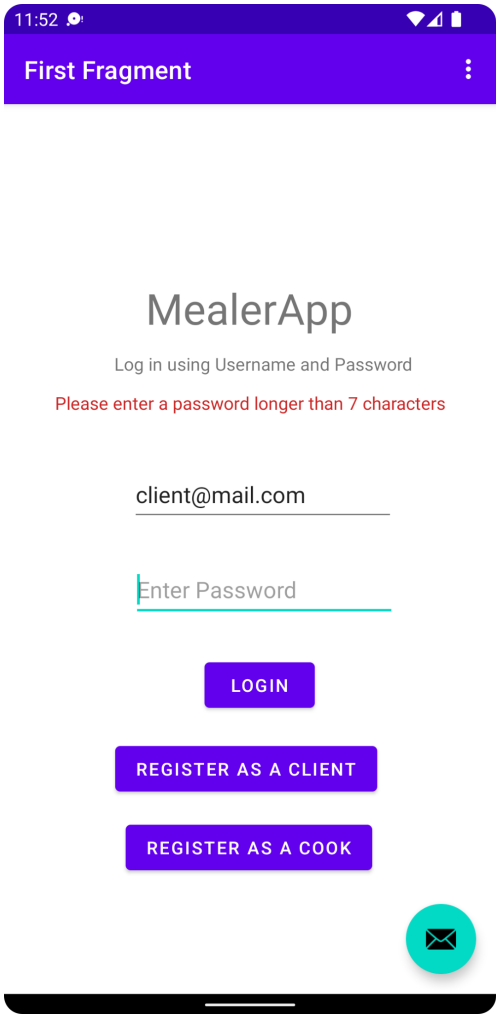
Screenshots



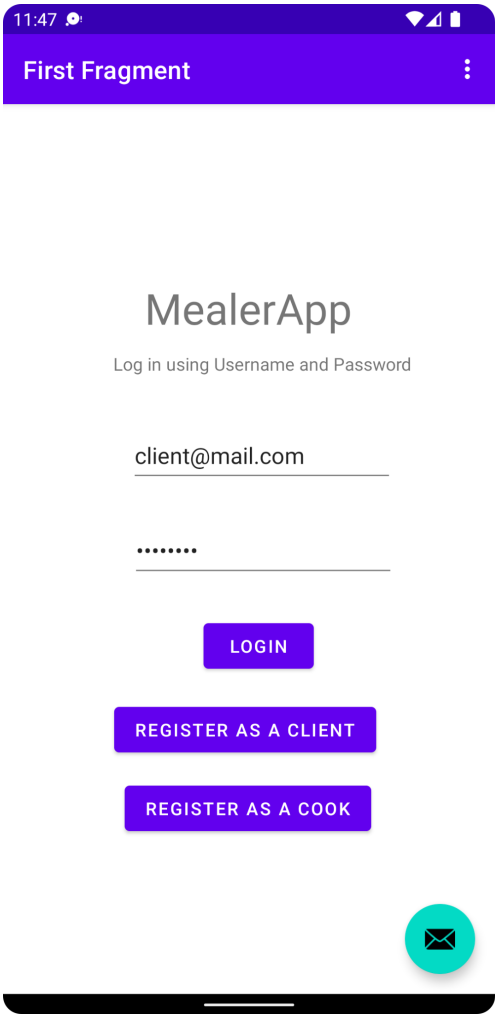
Landing screen of the app



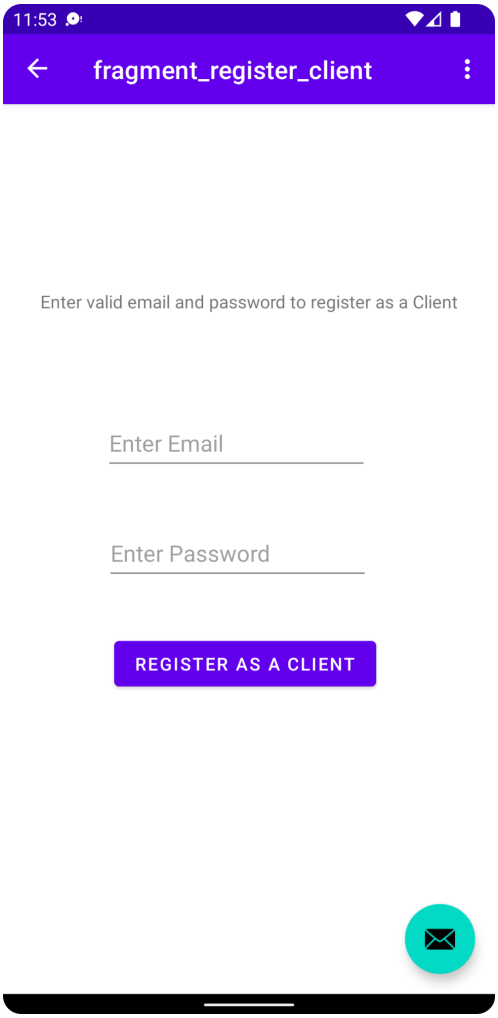
Invalid email and password message when user attempts to login



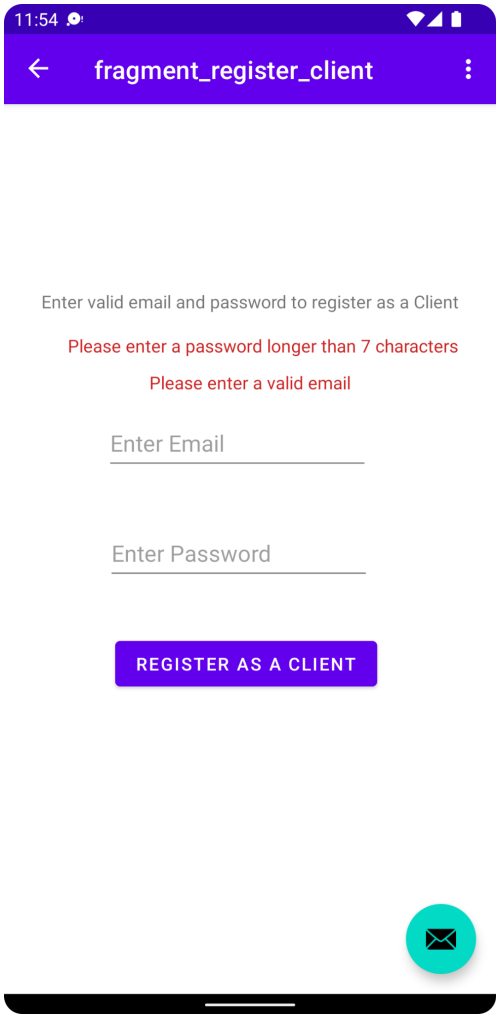
Invalid password message when user attempts to login



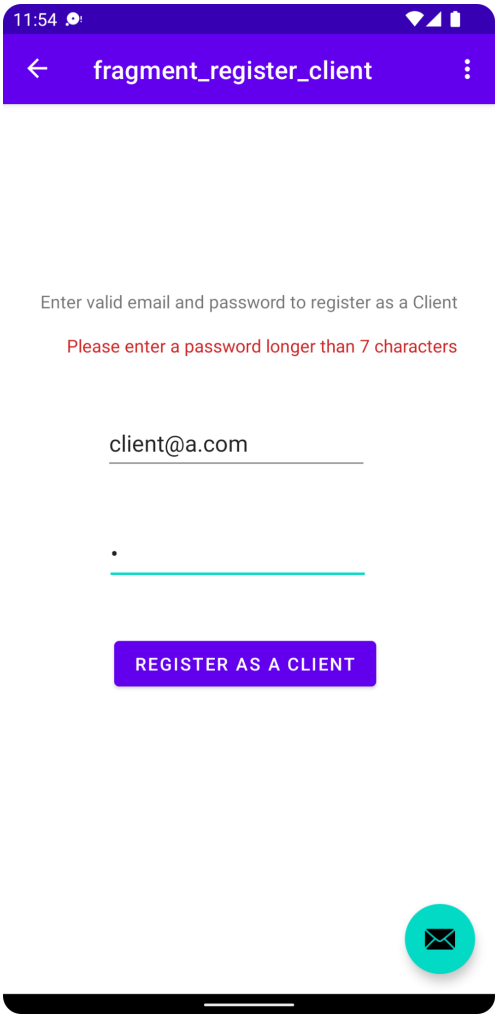
Initial screen with proper inputs



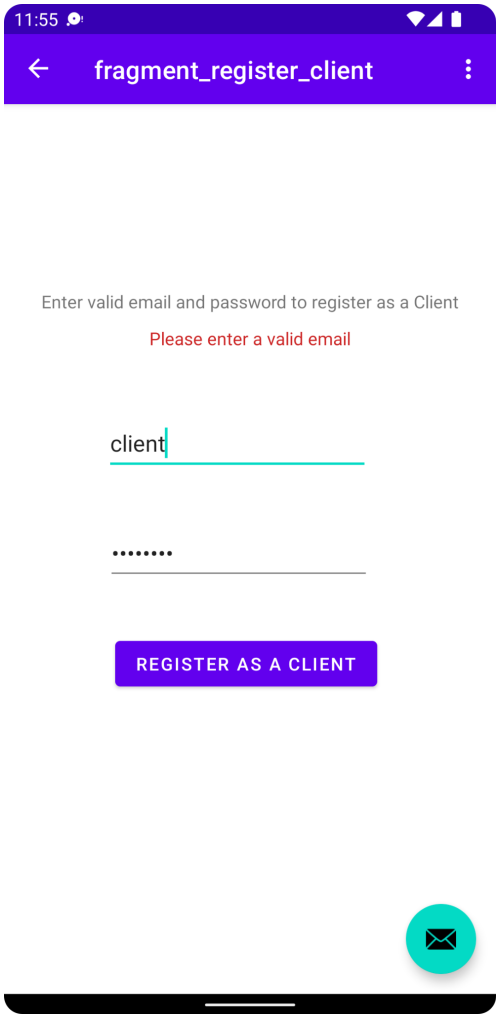
Register for client screen



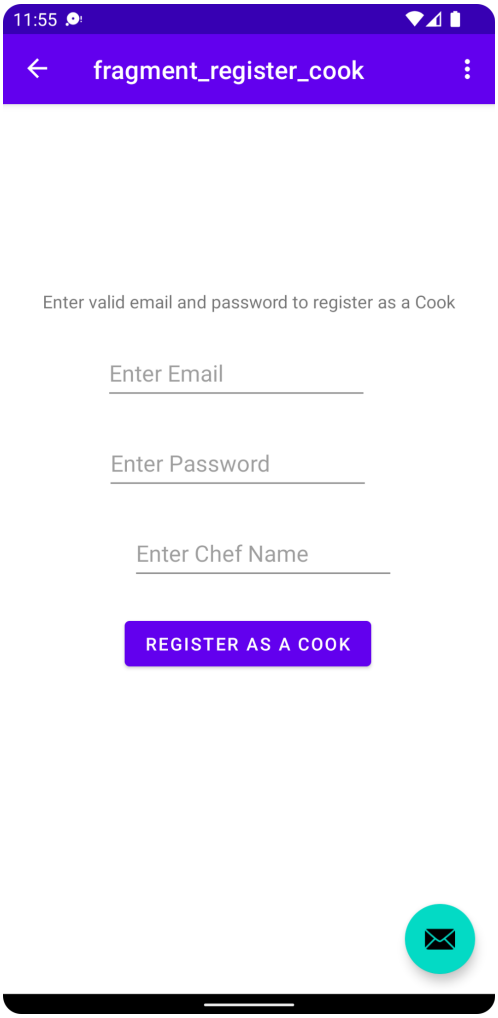
Register for client screen with invalid input message



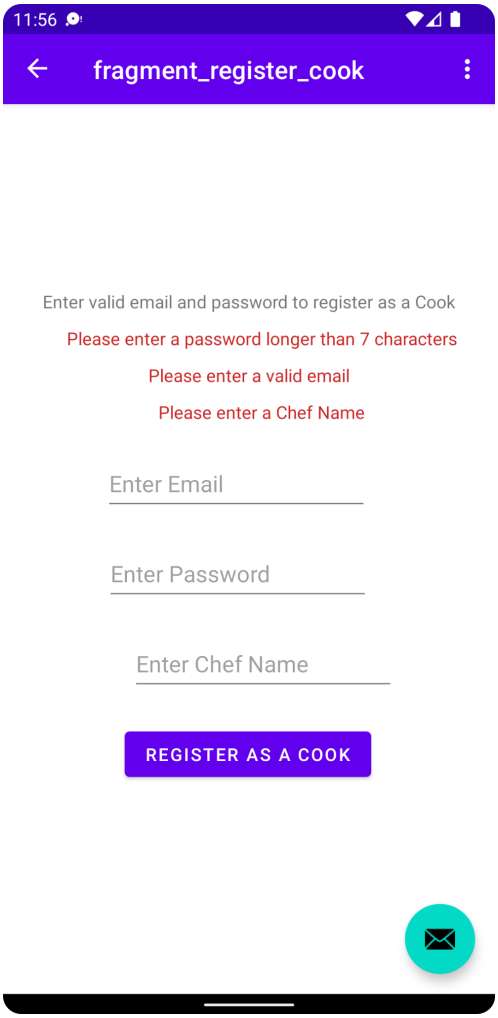
Invalid password message when signing up as client



Invalid email message when signing up as client

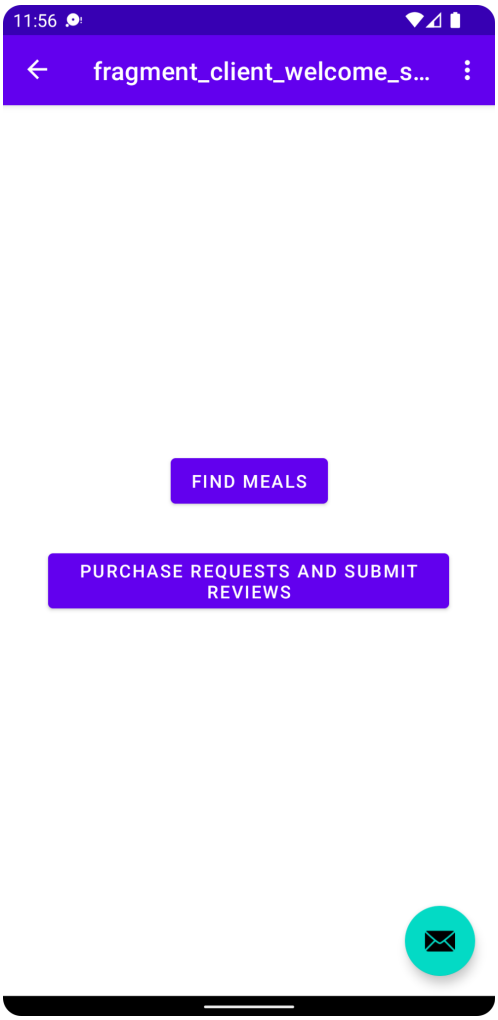


Registering as cook page

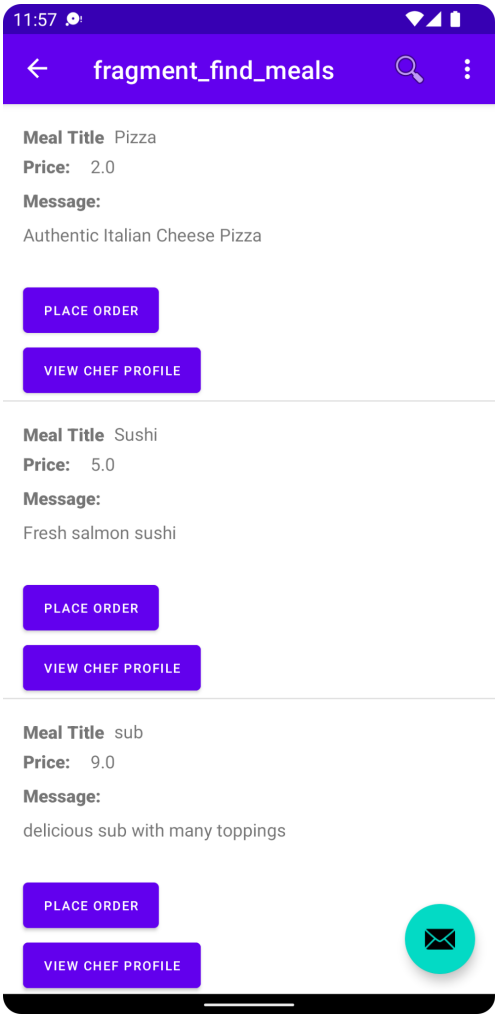


Invalid input messages shown on cook register page

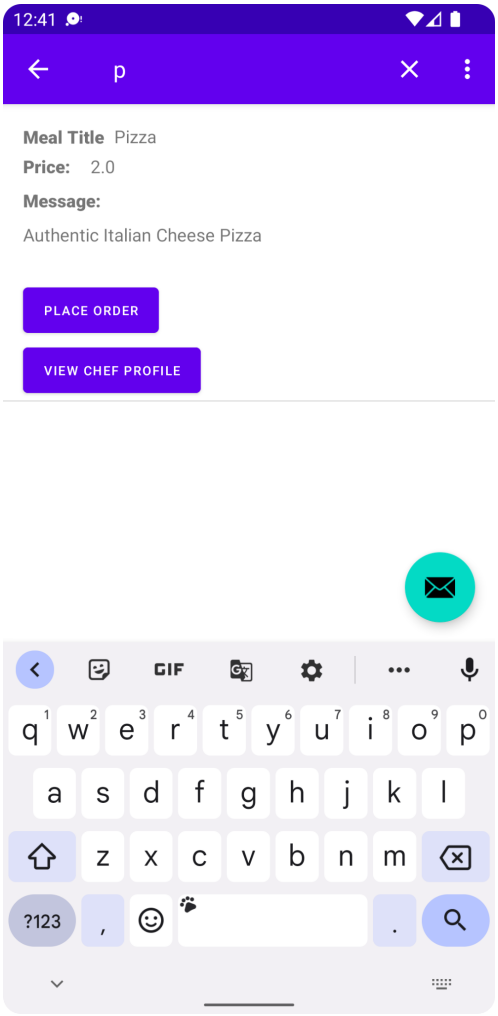
Client Screens:



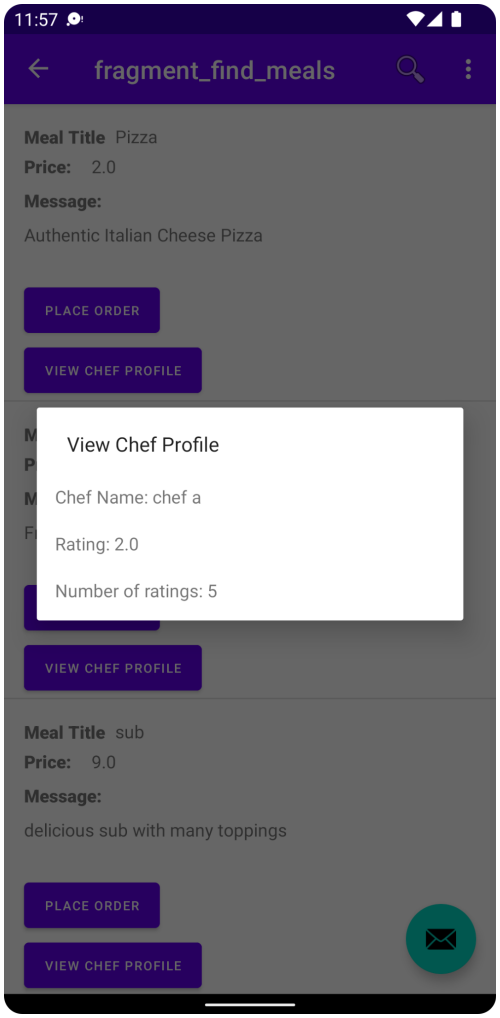
Client welcome screen



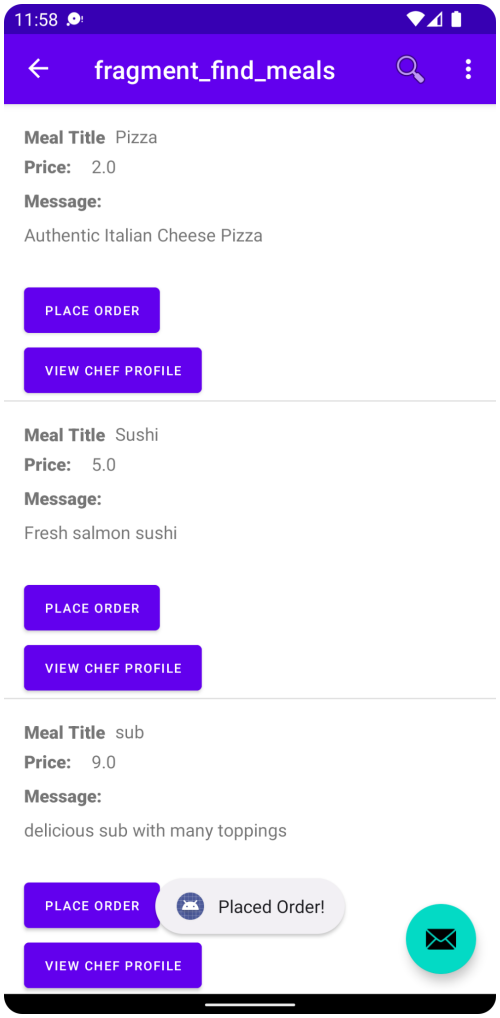
Client find meals screen that only displays menu items from non-suspended users



Client find meals search functionality

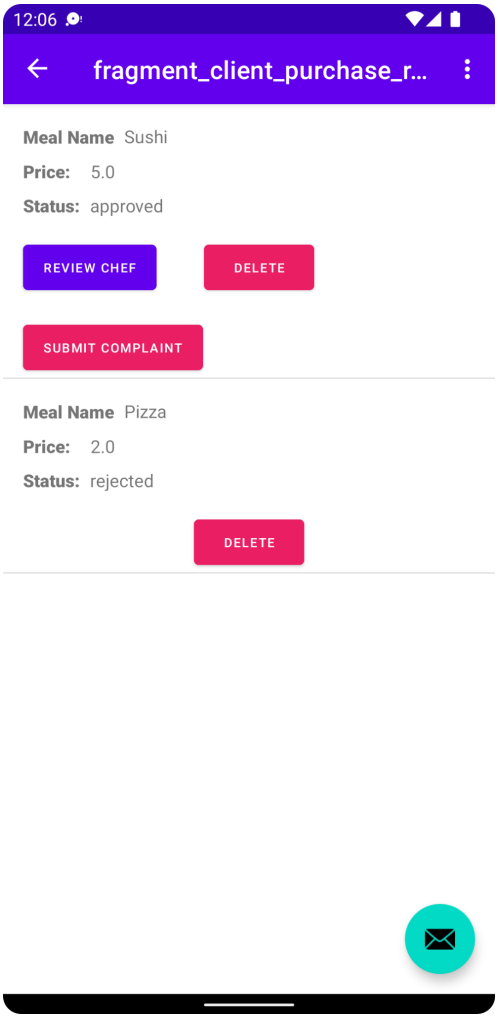


Viewing the chef profile for a meal

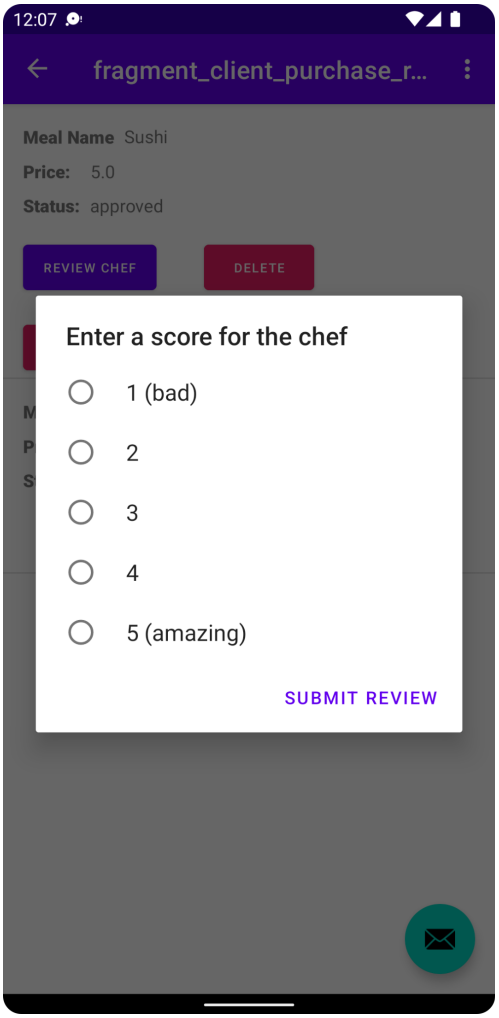


Success message when client orders a meal

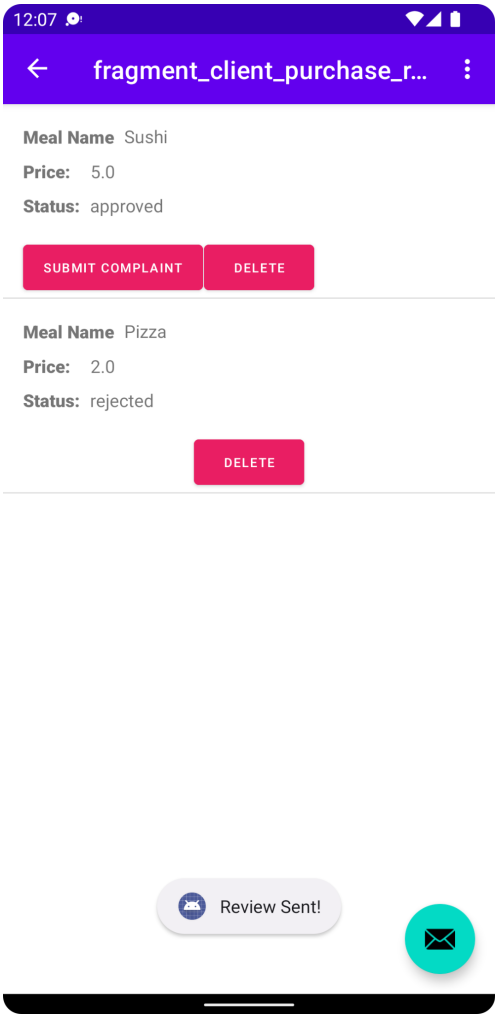
Client purchase requests screen with pending requests



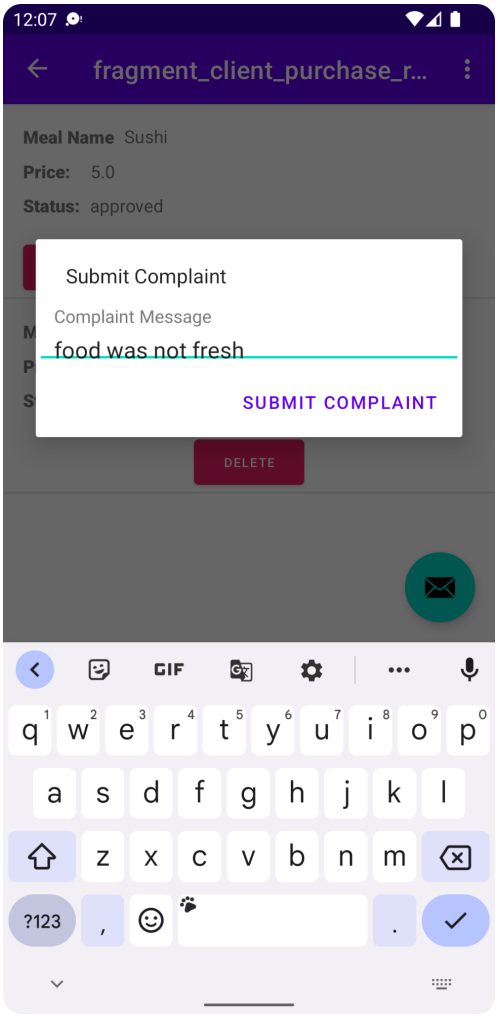
Client purchase requests screen with an approved and rejected requests



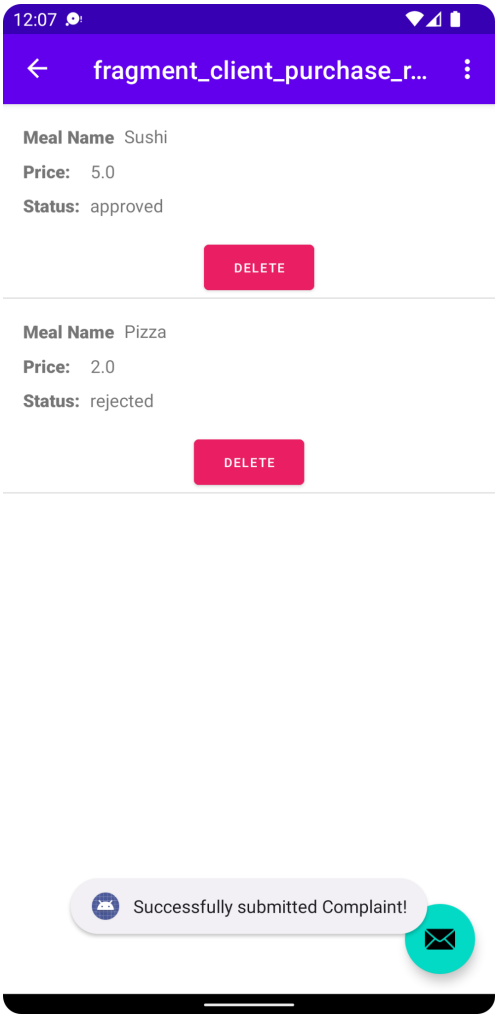
Client entering a score to review the chef for the approved meal



Success message when client sends a review to the chef

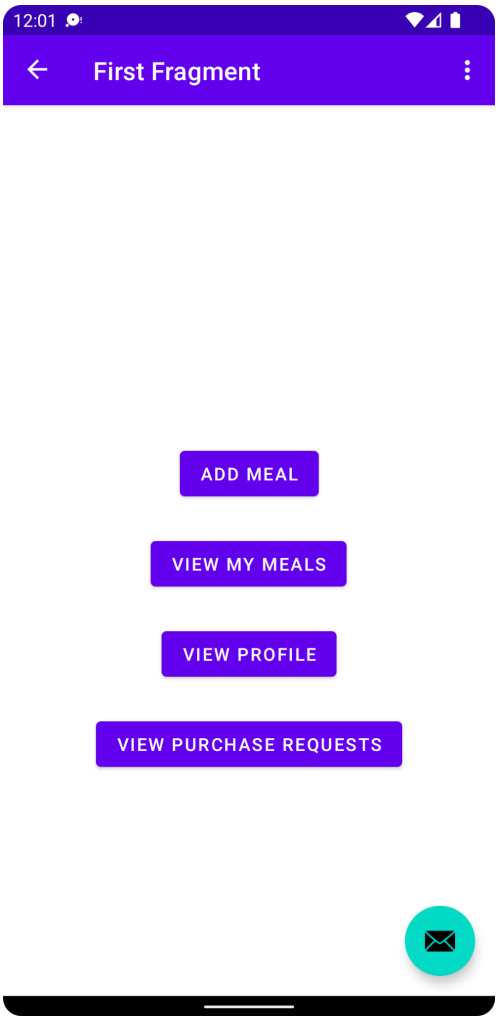


Dialog box when client submits a complaint

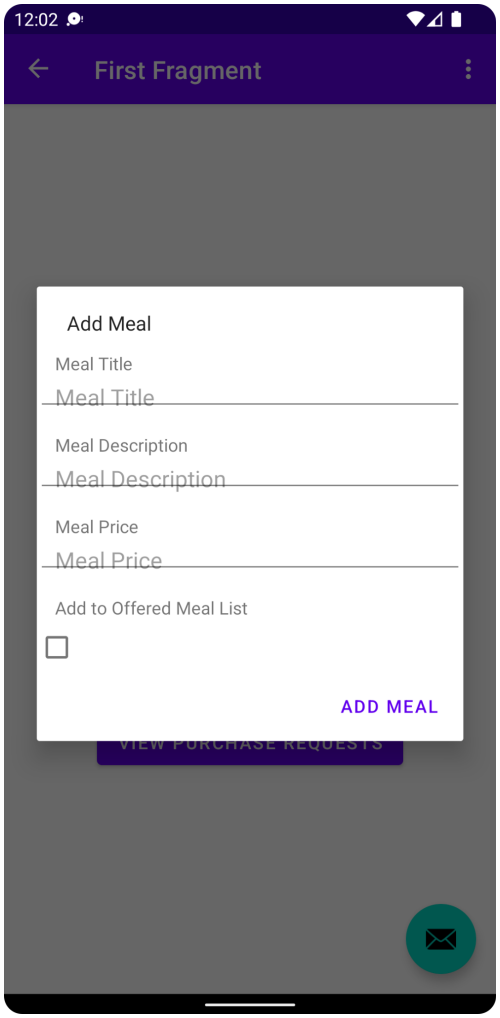


Success message when a user submits a complaint

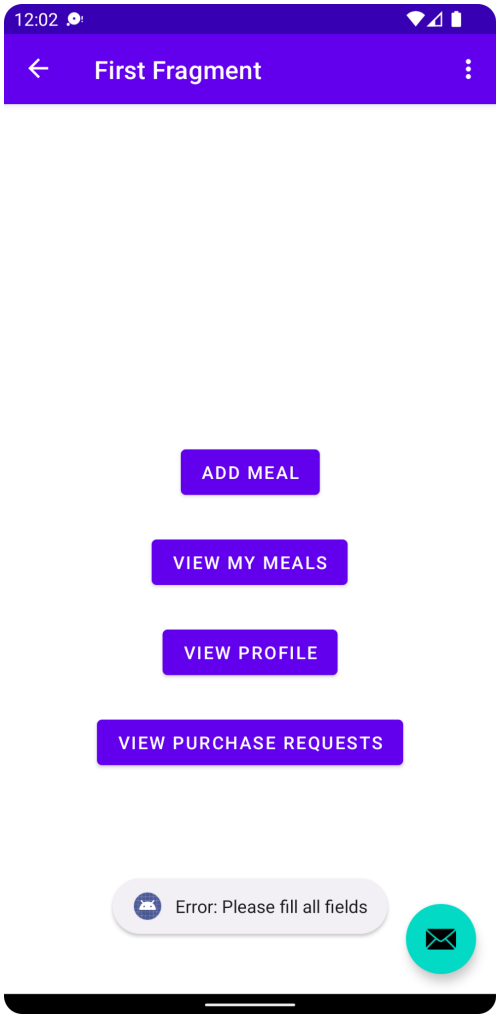
Cook Screens:



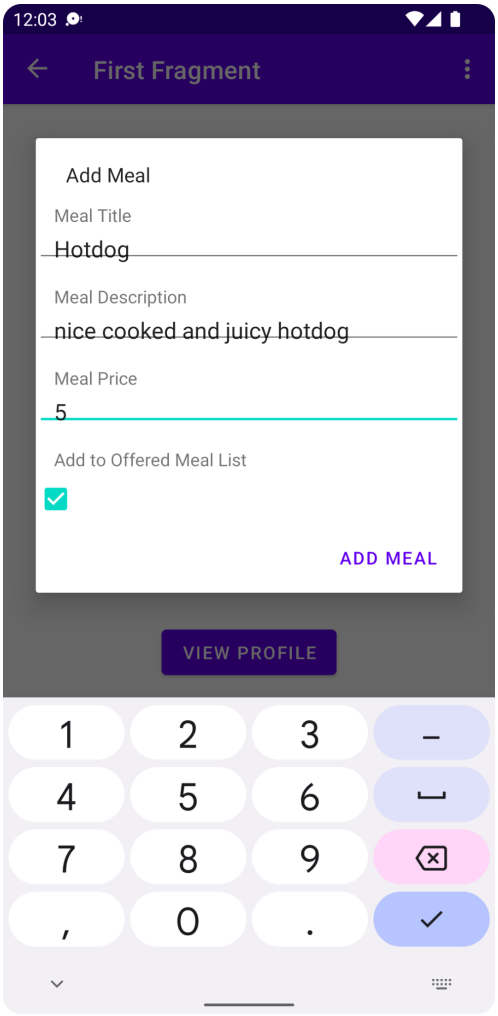
Cook welcome screen



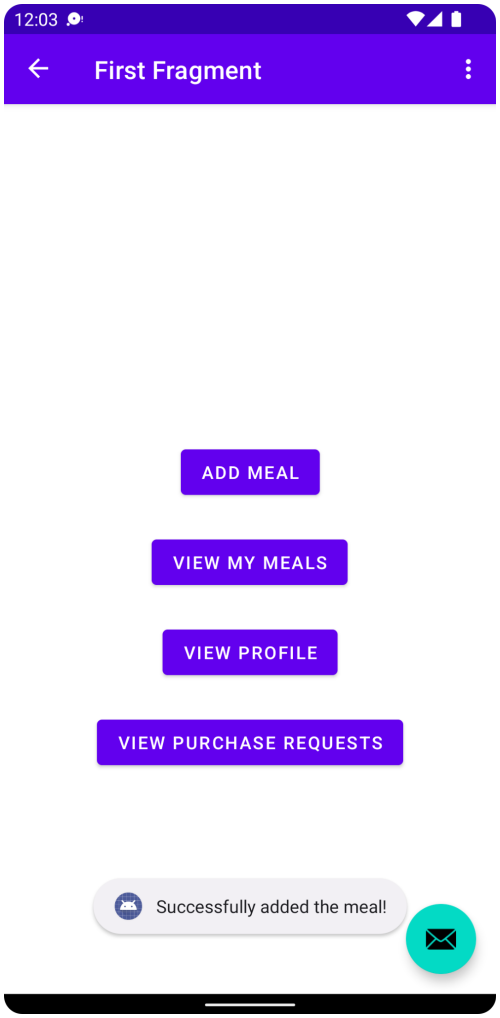
Dialog for cook to enter details about the meal



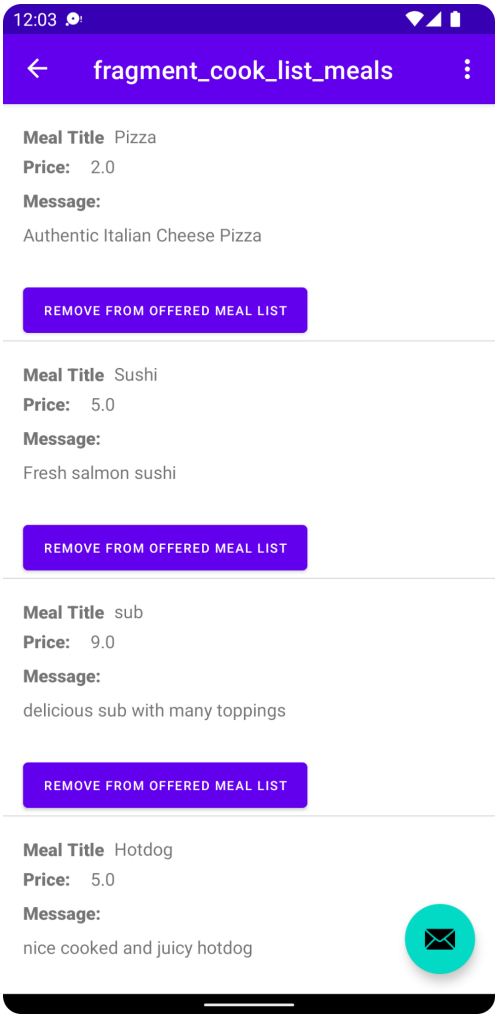
Error message if cook submits meal with missing information



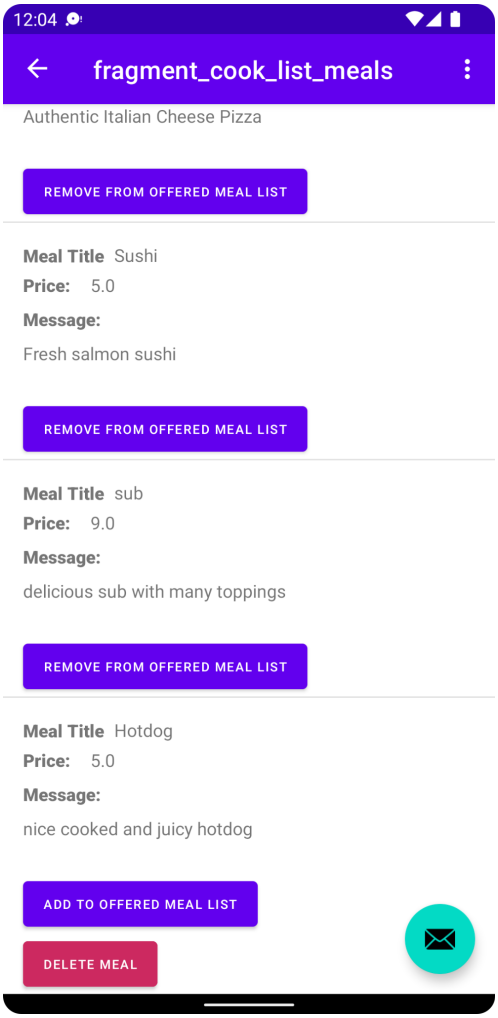
Submitting meal with completed information



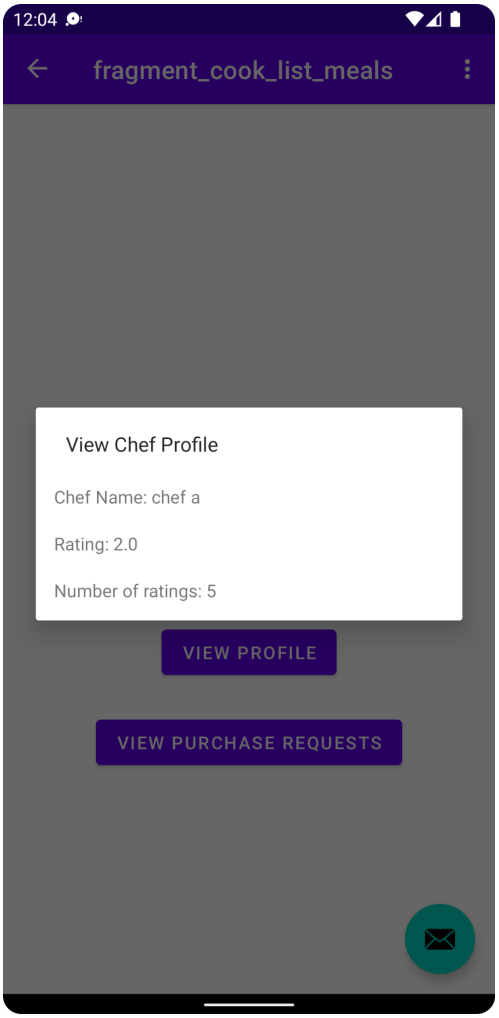
Success message when cook adds a meal



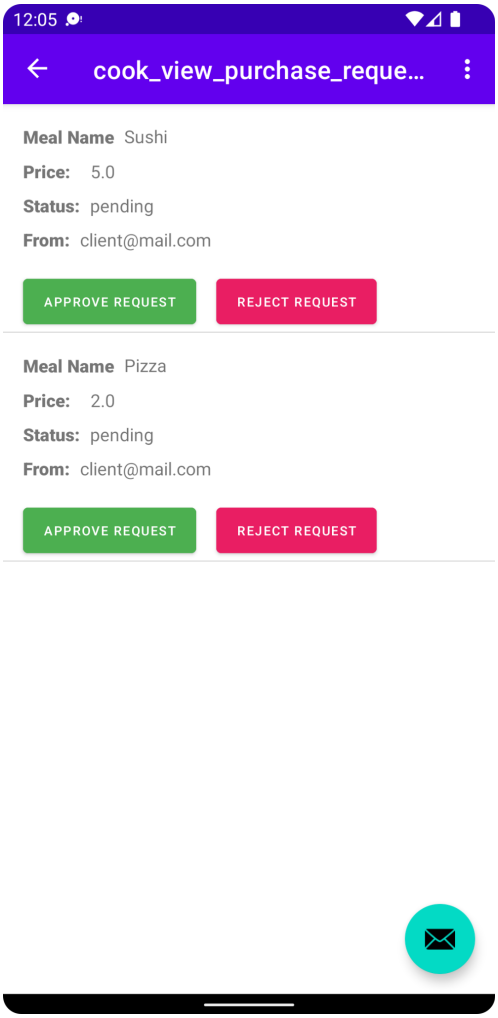
List of meals that belong to the cook



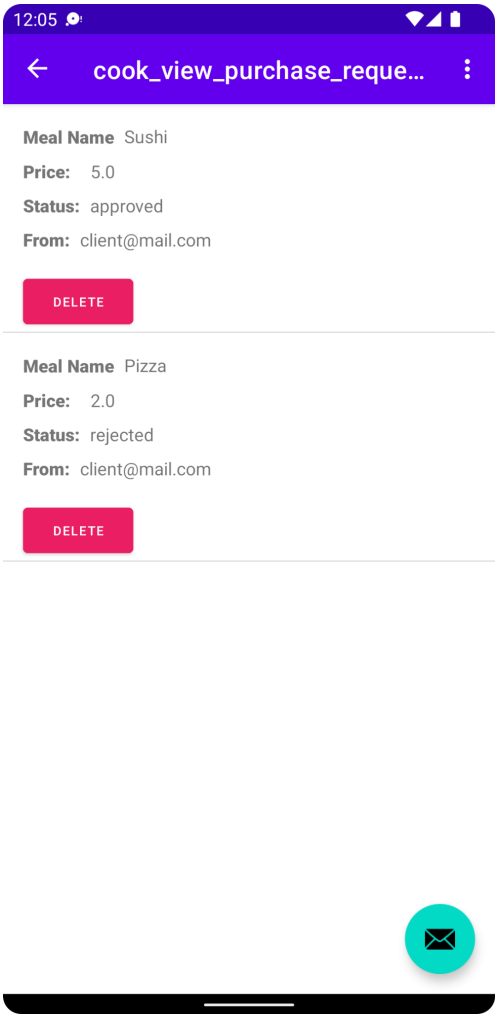
Optionality for the cook to delete a meal when it is removed from the offered meal list



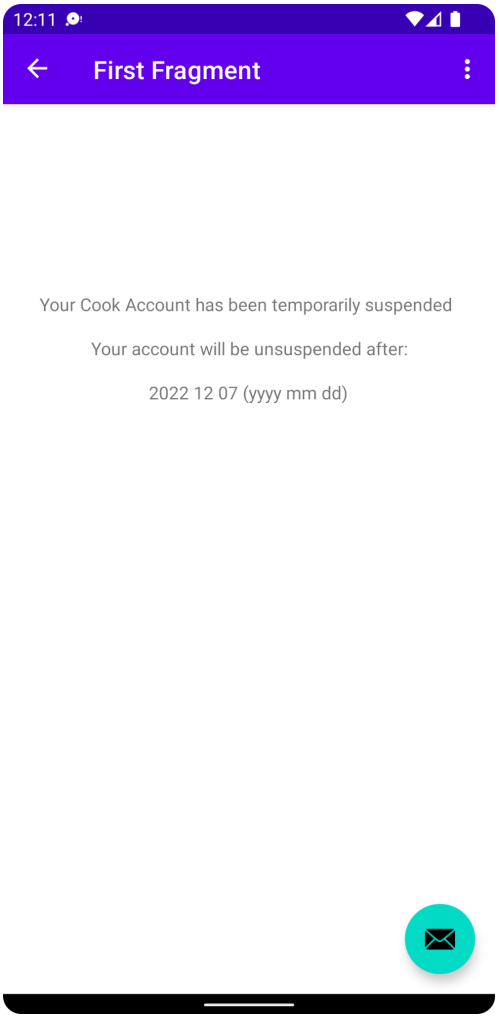
Dialog that appears when cook presses ‘View Profile’



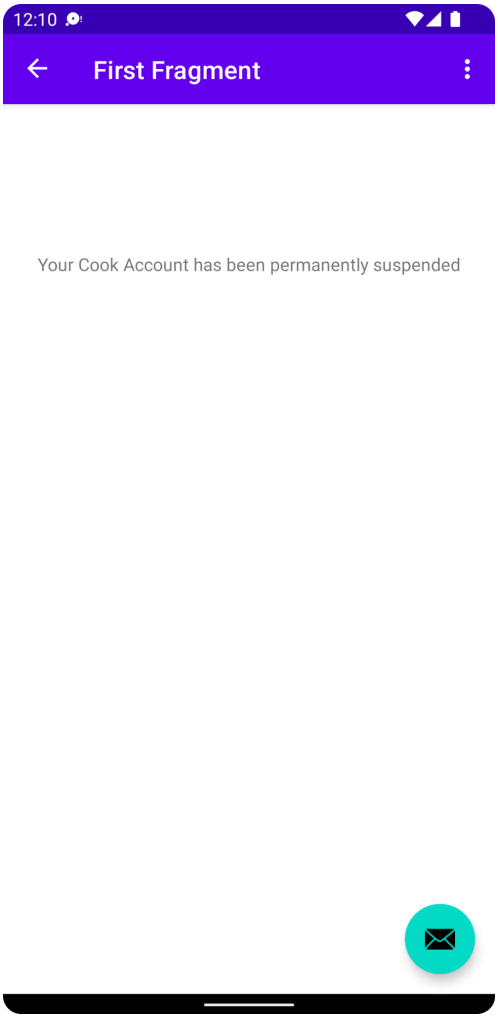
Screen for cook to manage purchase requests



Screen when the cook approves and rejects a purchase request

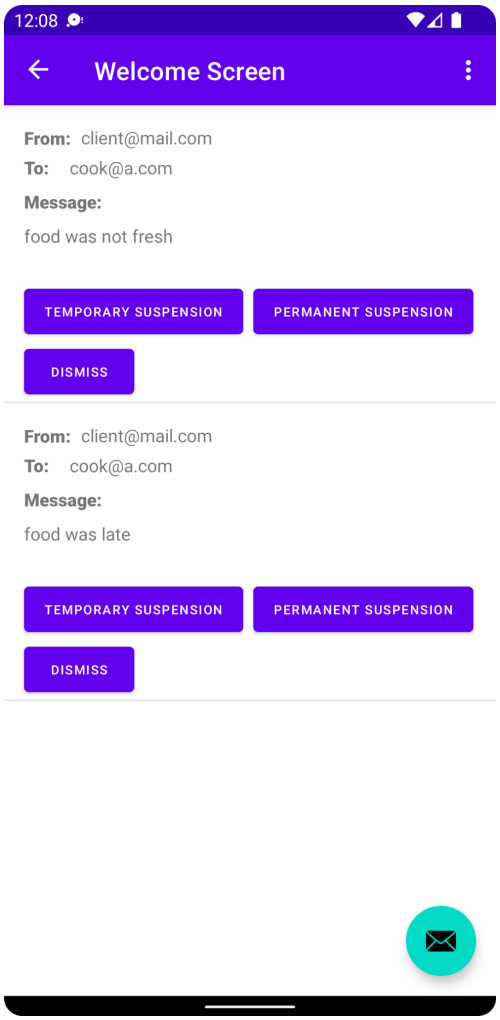


Cook temporarily suspended message upon logging in

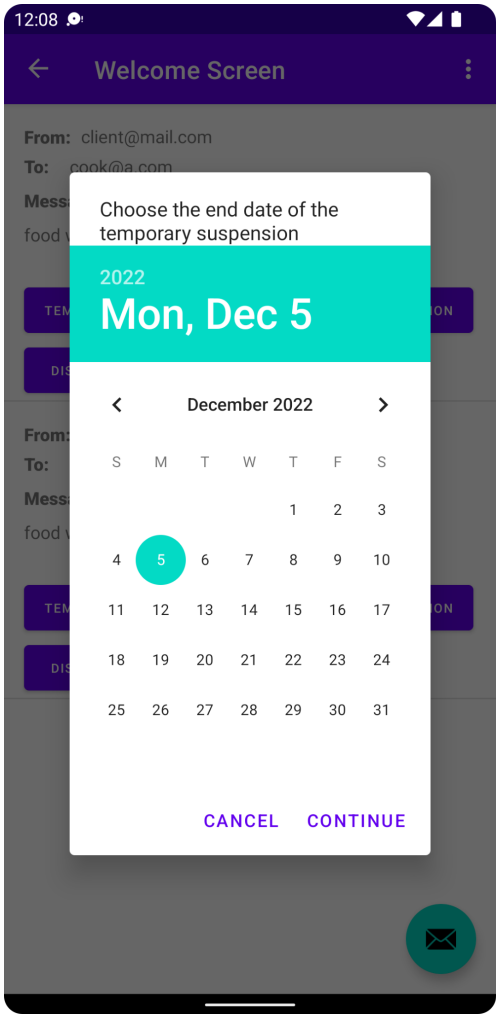


Cook permanent suspension message upon logging in

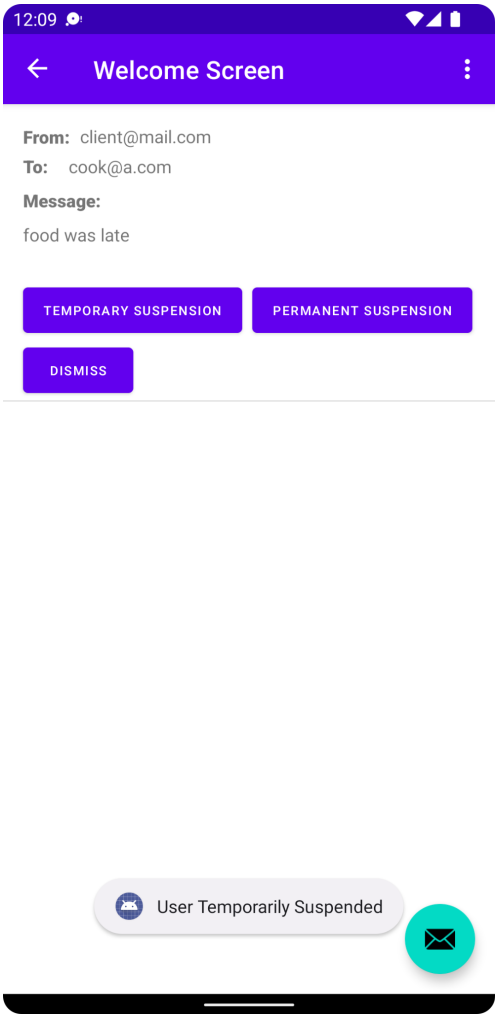
Admin Screens:



Admin welcome screen that contains the list of complaints



Date picker that appears upon pressing the `temporary suspension` button



Success message upon temporarily suspending cook

Success message upon permanently suspending cook

Lessons Learned

The following is a list of the most important lessons we learned as we submitted each deliverable:

- When creating an app, the mindset should be that the app could one day scale, and the data types should be receptive to change as well. As we completed more deliverables, we made changes to the UML diagram and implementing these changes in the code was easy since every object was defined in its own class within our project.

- It is important to build the initial features of the app as robust as possible, because as the project gets more sophisticated, it will be more difficult to add more features. However, if the initial features were created in an organized manner, creating more features will create less toil.

- There is always more work to do on an app to make it ‘perfect’. Whether it is optimization, error handling, UI design, code organization, an app can always be made better. We learned to balance between getting the objectives done and building the app in a maintainable way.

- Although it is easier to create features in a quick and unorganized way, it is much more sustainable to Don’t go for the ‘quick’ fix and rather do the proper fix to prevent other issues down the line