The Cougar-Devs - Anil Chandler, Nima Barahmi, Gabriel Trezza
Professor Hardie
CSCI 360-01
December 7, 2022

Final Software Architecture Document

== Table of Contents ==

== Glossary ==

Items - Product of service for sale
Tickets - Electronic form of entry to specified show

== Vision ==

Our vision is to create a ticking platform to be used campus wide! This platform will be used largely with performances on campus, but hopefully for more as well. Users will be able to buy tickets at associated events with the College of Charleston. The mobile app will be able to run on all mobile devices (Androids, iPhones, etc.). The estimated time of delivery of the final product will be December 2023.

== Use Cases ==

USE CASE #1 - Purchasing a Ticket

Scope:

CharlestonEvents application

Level:

User goal

Primary Actor:

User

Stakeholders and Interests:

User: Wants to be able to purchase the ticket with minimal effort. Wants proof of purchase to confirm ticket purchase. **Government Tax Agencies:** Wants to collect tax from every sale. May be multiple agencies, such as national, state, and county.
Company: Wants to be able to update the events available such as adding or deleting an event.
Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.

Preconditions:

User is logged in and must have a form of payment, User already selected an event from events.

Postconditions:

Users are able to purchase a ticket. Taxes are correctly calculated. Receipt is generated.

Main Success Scenario:

1. User logs in to the application
2. User finds the event
3. System presents total price with taxes calculated
4. User pays and System handles payment
5. System updates the amount of tickets available

Extensions:

A. The user is not logged in: User finds the event .(used case moved to browsing section) System presents price without taxes being calculated User tries to purchase the ticket System redirects the user to the login page

B. The event has no available tickets: User searches for the event User finds the event System informs the user that there are no more tickets available

C. The user leaves the application before confirming purchase: System saves the last page visited by the user When the user returns to the application, system displays the last page the user had visited

D. User method of payment has failed: User inserts their method of payment The system submits the payment The form of payment has not been accepted System informs the user System tells the user to try another method of payment If another payment method has been inserted: Repeat step 2 If payment method has been accepted: System emails the user the receipt of payment If payment method has not been accepted: Repeat step 3 to 5

Special Requirements: User must be at least 18 years of age to purchase a ticket Credit authorization response within 30 seconds 90% of the time. Must be completed within the Fall Semester of 2022

Technology and Data Variations list: The application must run on IOS Platform and Android Platform

USE CASE #2 - Browsing Shows

Name:

Browsing Shows

Scope:

Browse events

Level:

User goal

Primary Actor:

User

Stakeholders / Interest:

User: The user wants to browse and see all events in Charleston, and be able to select an event and proceed on ticket purchasing window.

Event organizer: The event organizer could list all events for the following months and may do such a thing by requesting developers to update the app, events can be added or removed by the developer team only upon request form the event organizer .

Preconditions: User is logged in and must have a form of payment to be able to view the events.

Postconditions:

User has browsed all events and selected the one particular event, and moved on to the purchasing window.

Main Success Scenario:

User searches for an event . User finds the event . User can chose the event User can see a brief description of the event such as date and title. Selected events will be added to the purchasing basket(window).

Extensions:

A. The user is not logged in: User searches for the event . User finds the event . User may not be able to move on to the purchasing pos.

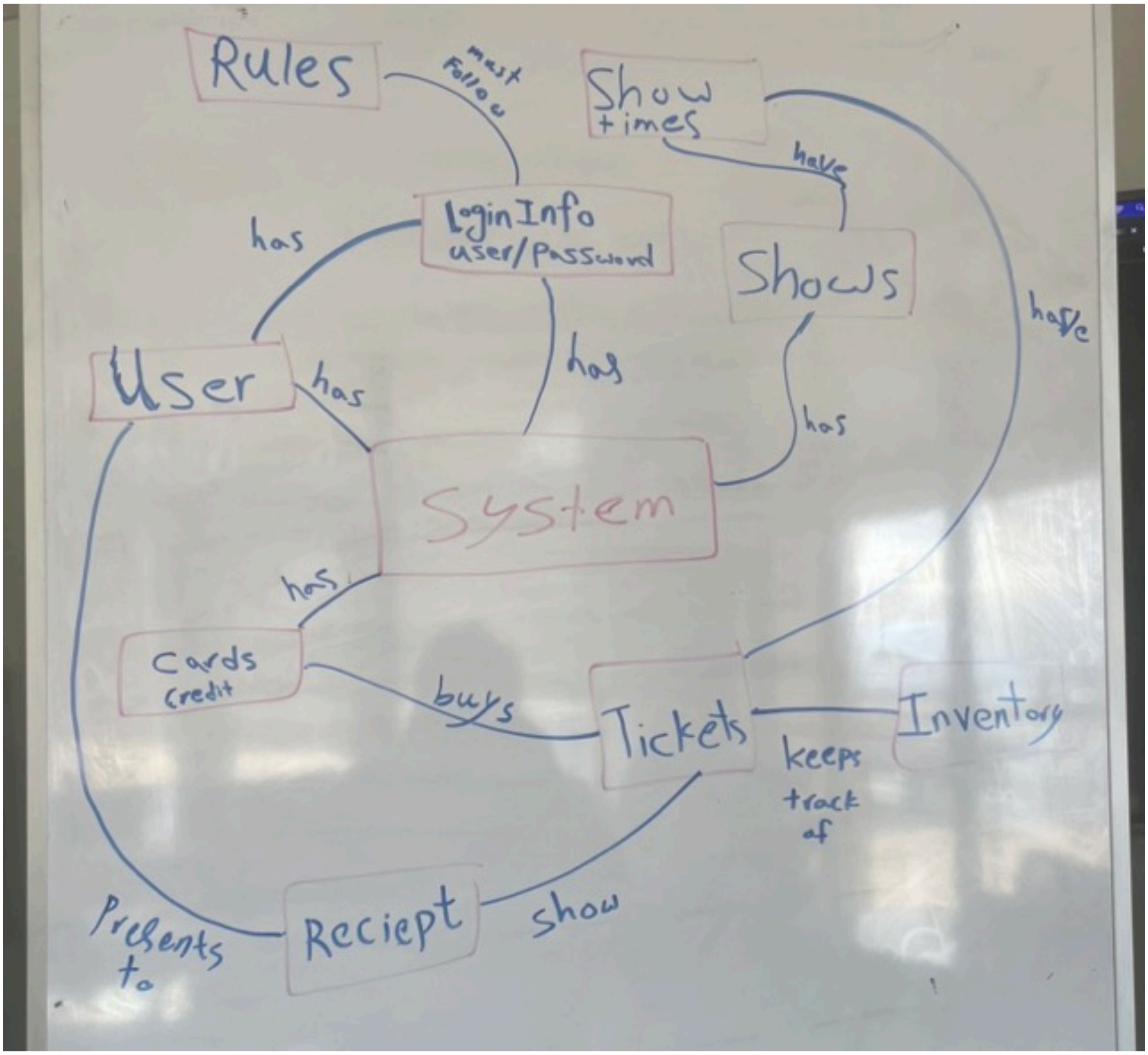
B. User selected expired/past events: The user will be notified by an alert that the event does not exist or expire. User can dismiss the alert and browse other events.

C. The user is logged in: User may chose to browse events. The events must not be expired. User may select an event and come back from purchasing terminal, without actually purchasing any ticket.

D. No event exists: In case no event exist, event page/window notifies the user that currently there is no available events in Charleston. Special Requirements: - The UI must be user friendly. - The font and UI elements must follow Apple/Android interface guidelines. - The event titles must be bold - dates and description of the event must be smaller than the event title.

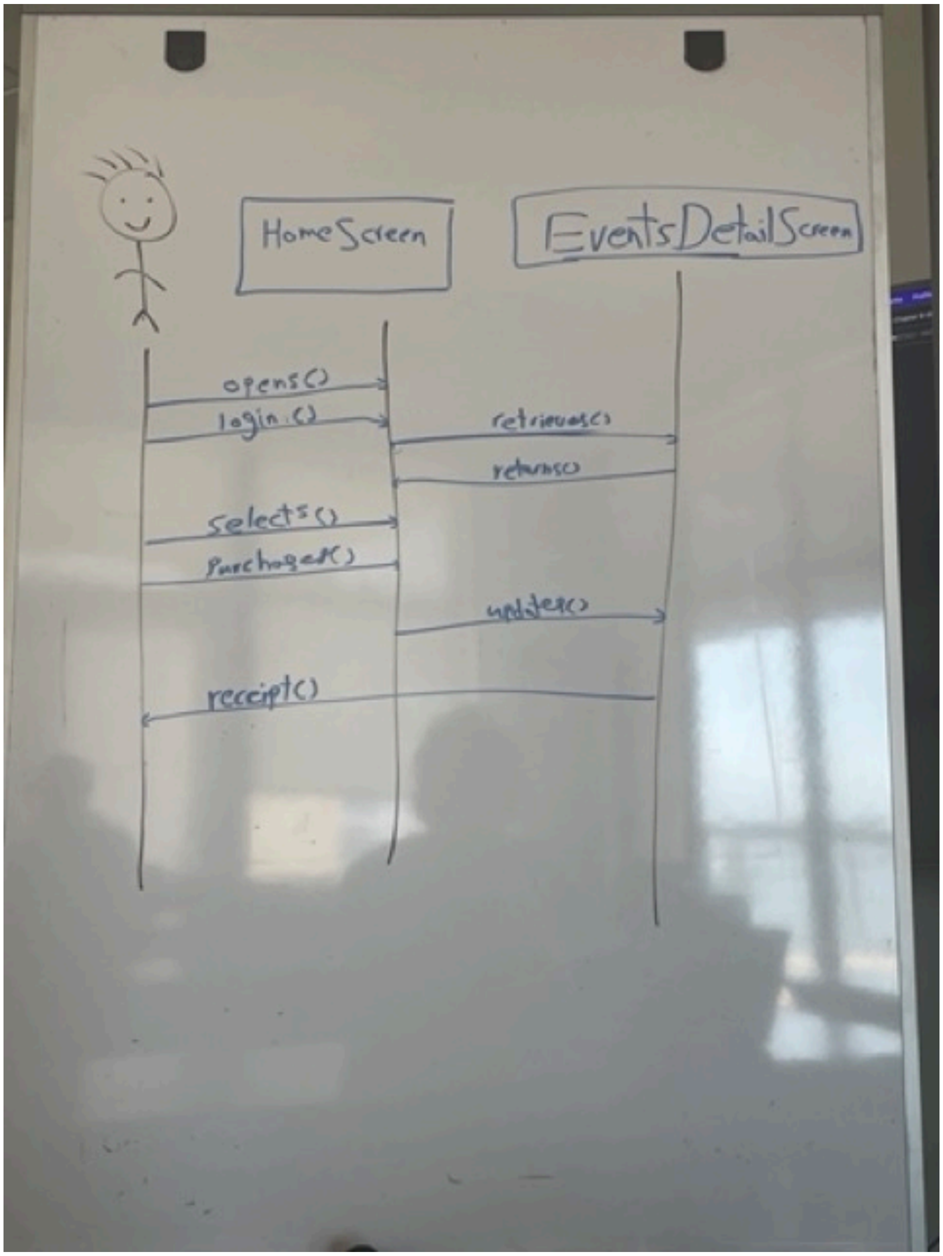
Technology and Data Variations list: The application must run on IOS and Android Platform.

== Domain Model ==

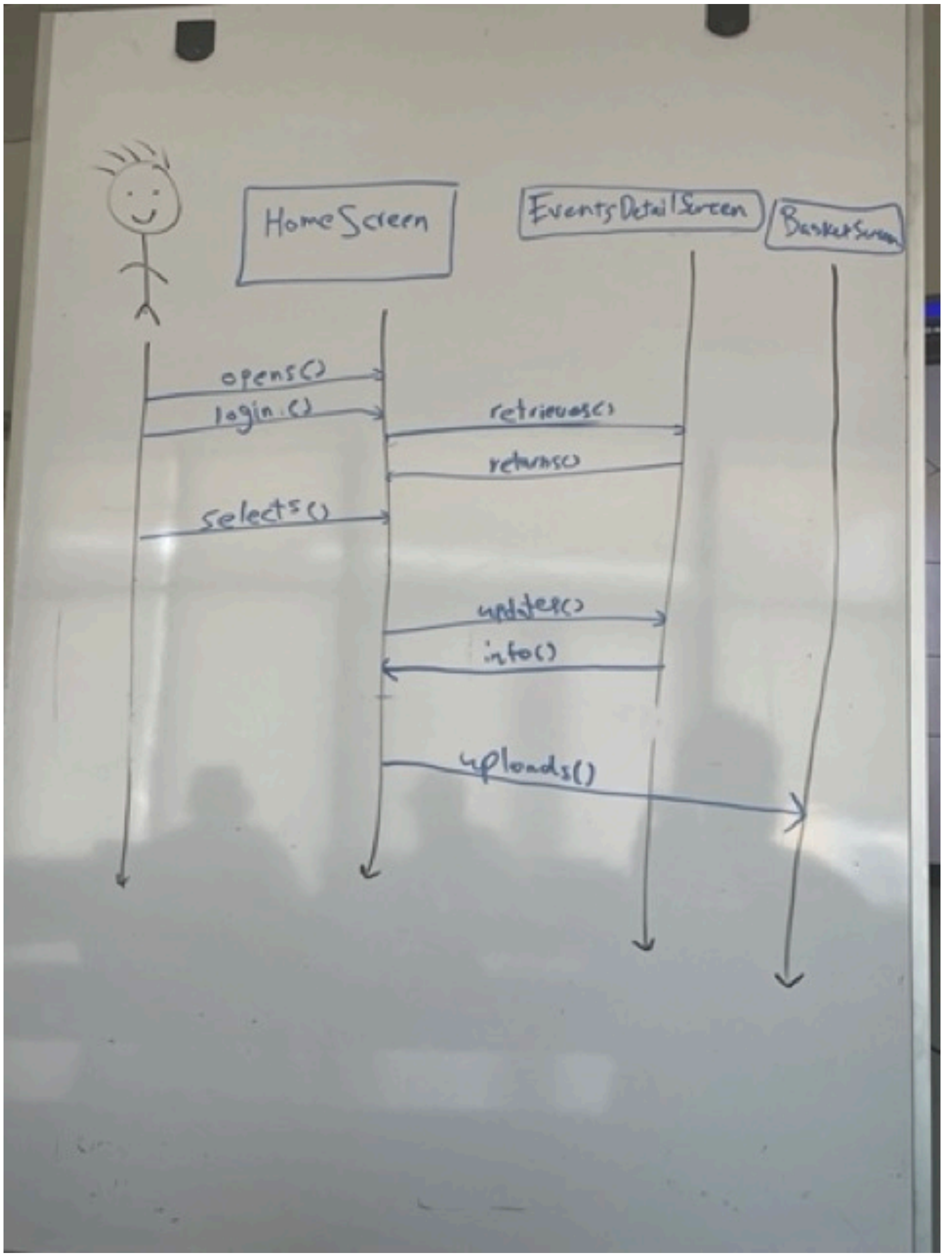


== System Sequence Diagrams ==

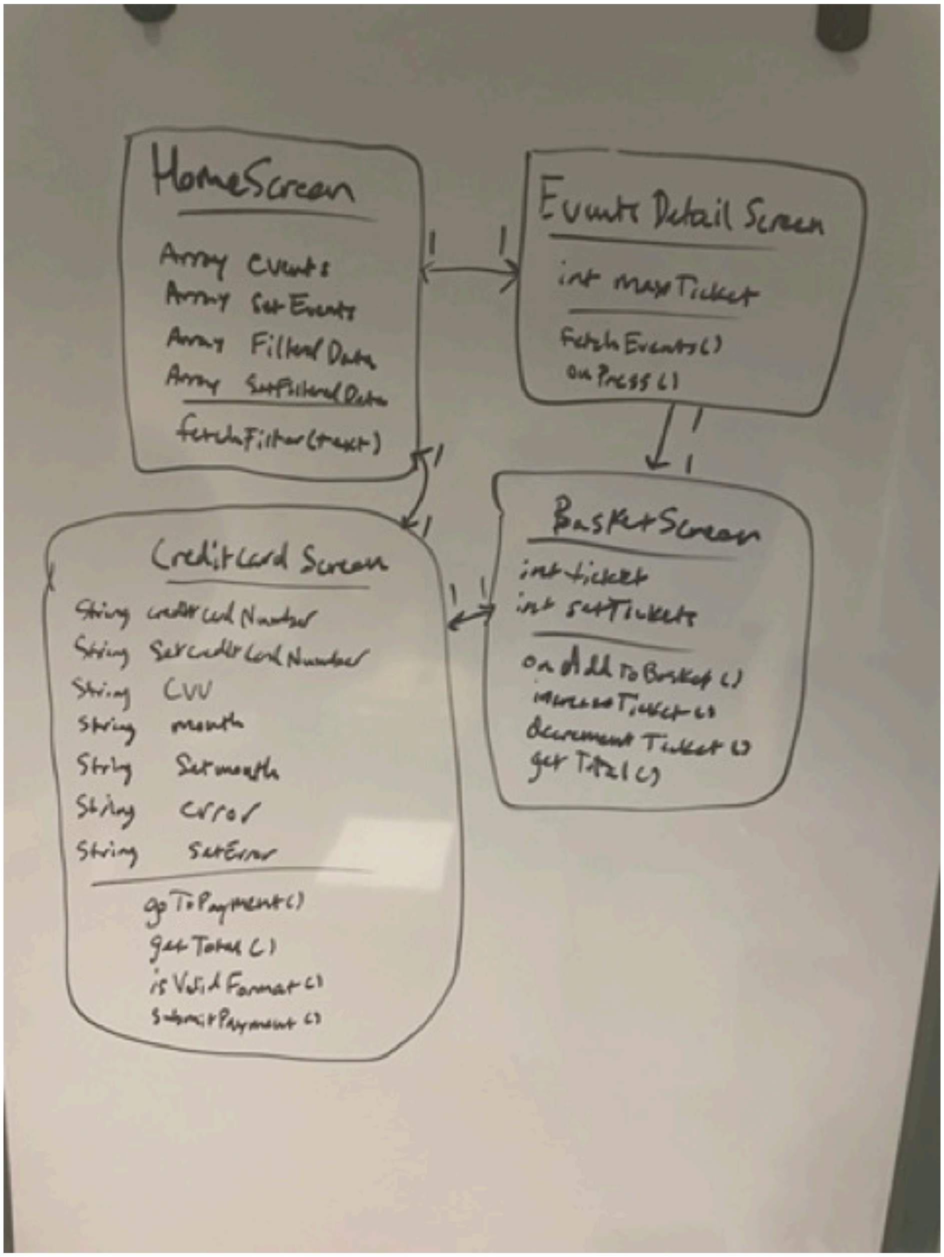
== Purchasing Ticket ==



== Browsing Shows ==



== Class Diagrams ==



== Operation Contacts ==

Operation Contract #1 - Purchasing Tickets

Operation: opens()

Cross References: None

Preconditions: None

Postconditions: First View Presents

Operation: login()

Cross References: Use Case: Not Defined

Preconditions: None

Postconditions:

- A SalesLineItem instance was created (instance creation).
- The App associated with the current user (association formed)

Operation: searching()

Cross References: Use Cases: Browsing

Preconditions: logged In

Postconditions:

- Event instance was created (instance creation)
- Display certain amount of events (attribute modification).
- Event was associated with a Event Details, based on itemID match (association formed).

- A Payment instance p was created (instance creation).
- p.amountTendered became amount (attribute modification). – p was associated with the current user formed).
- The current Sale was associated with the event organizer (association formed)

Operation: purchasing()

Cross References: Use Cases: Purchasing

Preconditions: User is browsing events

Postconditions:

- A Payment instance p was created (instance creation).
- p.amountTendered became amount (attribute modification). – p was associated with the current user formed).
- The current Sale was associated with the event organizer (association formed)

Operation Contract #2 - Browsing Tickets

Operation: open()

Cross reference: none

Preconditions: none

Postconditions: none

Operation: login()

Cross reference: Use case not defined

Preconditions: The app has been opened

Postconditions:

- Login Instance “LGI” has been created for login operation.

- LGI has been associated with the User.

Operation: purchases()

Cross reference: Use case: Purchasing a ticket

Preconditions: User should be logged in

Postconditions:

- A payment instance “p” has been created
- p.TotalAmount became Total Amount

Operation: retrieves()

Cross-reference: User case has not been created

Preconditions: User must be logged in

Postconditions:

- An instance events has been created
- Events has been displayed in the app

Operation: updates()

Cross-reference: User case has not been defined

Preconditions: User must be logged in and Events needs to have modifications

Postconditions:

- An update instance “upd” has been created
- upd.TotalAmtTicketsLeft became Tickets amount

Operation: receipt()

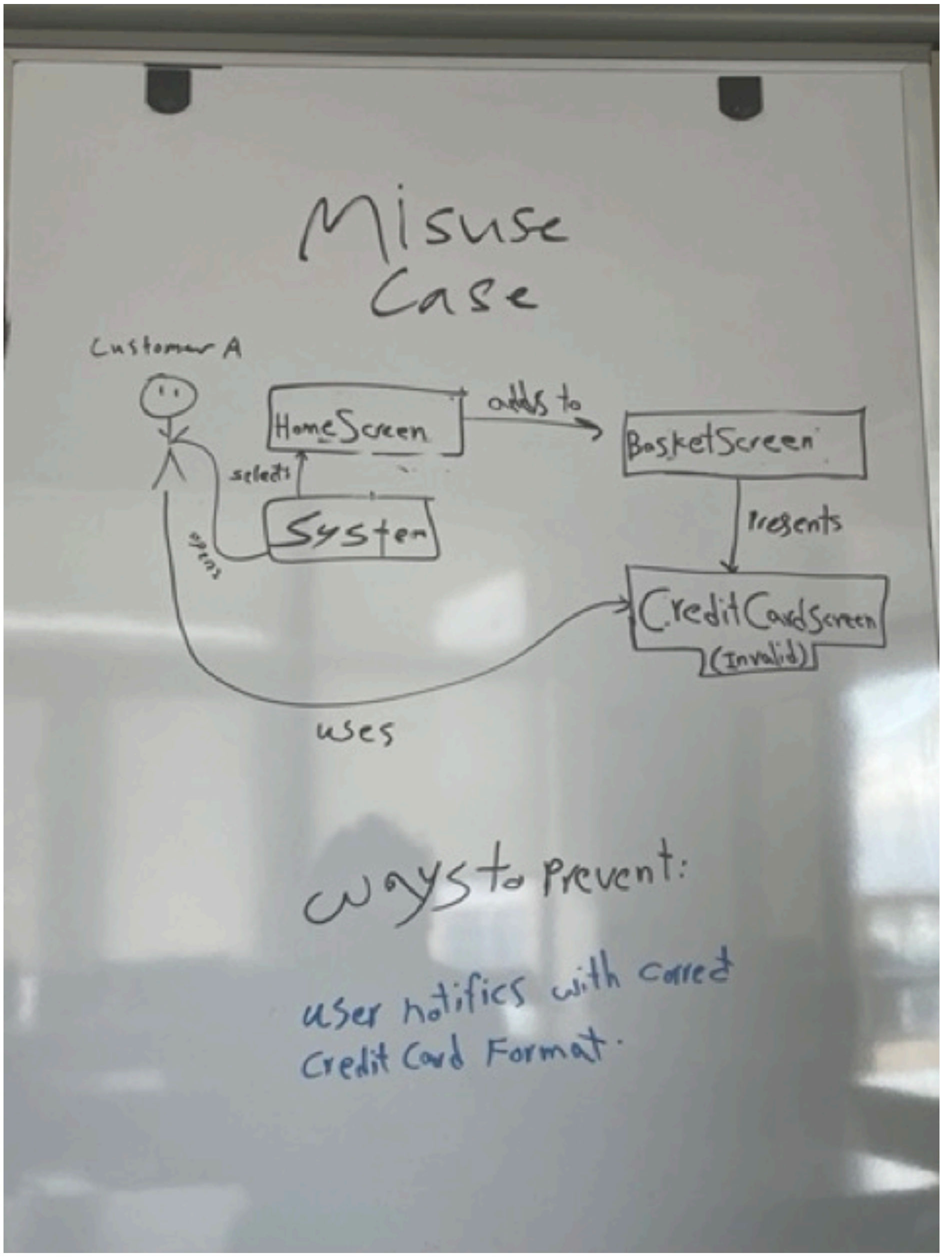
Cross-reference: Use case has not been defined

Preconditions: User purchased a ticket

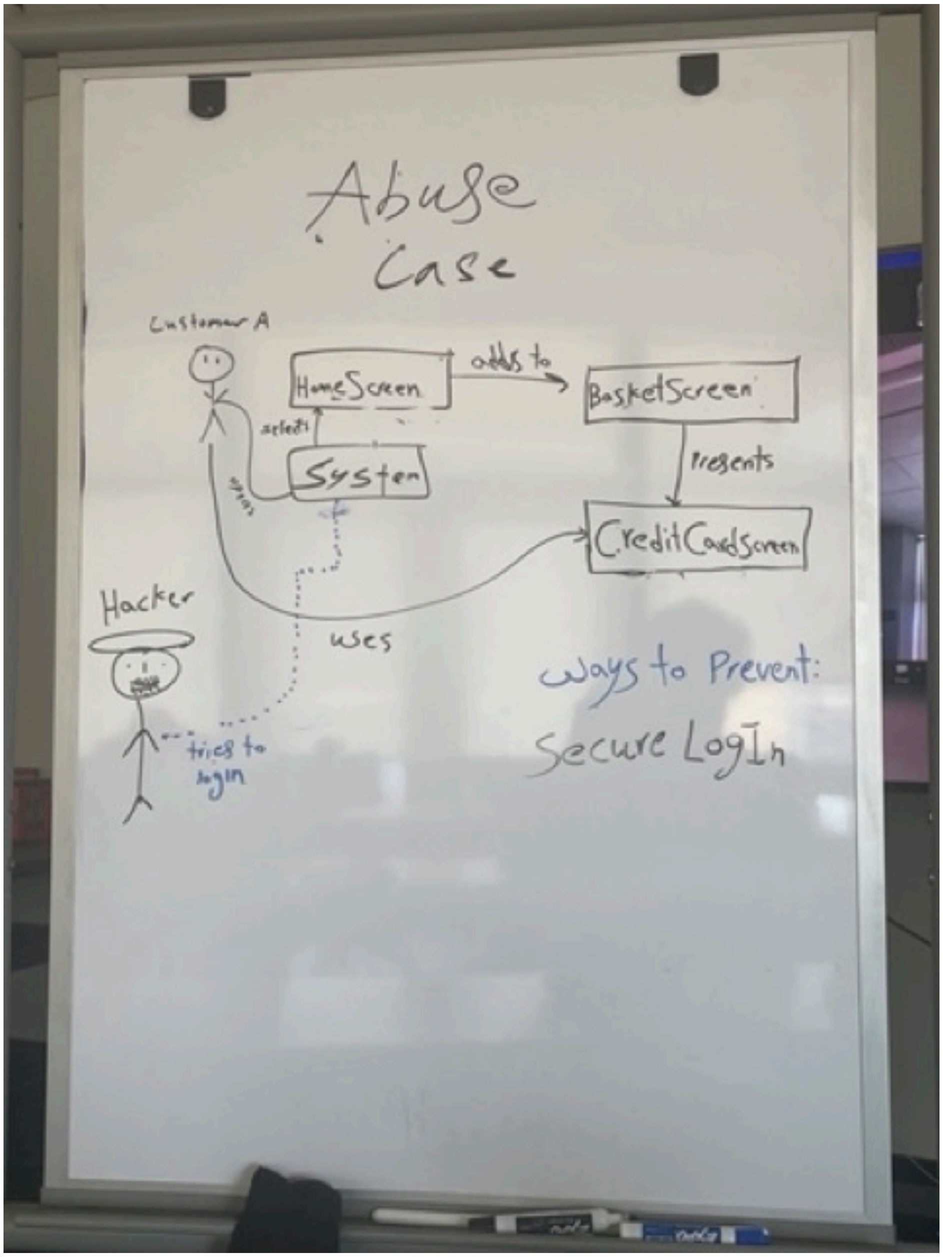
Postconditions:

- An alert is presented to with details of receipt

== Misuse Case ==



== Abuse Case ==


Abuse
Case
Customer A
HomeScreen
adds to
BasketScreen
selects
System
presents
CreditCardScreen
Hacker
uses
tries to
login
ways to Prevent:
Secure LogIn