CSC 431

ThriveStream

System Architecture Specification (SAS)

Team 07

Bianca Toto	Project Manager
Juan Arango	Developer
Sophie McIntyre	Scrum Master

Version History

Version	Date	Author(s)	Change Comments
1.0	4/1/21	Team07	First Draft
2.0	4/23/21	Team07	Second draft

# Table of Contents

Table of Figures

31. System Analysis

31.1 System Overview

The system will be made up of multiple parts: a user interface, a profile system server, a payment systems server, a verification component, and a library database. The system will utilize an Event-Driven architecture style, using a series of success or fail events that progress from one component to another. A series of successful events verify a user's account status, payment info, and ultimately begin streaming the event. Failure events will lead to error until the user fixes their information.

31.2 System Diagram

Figure 1.2 System Diagram
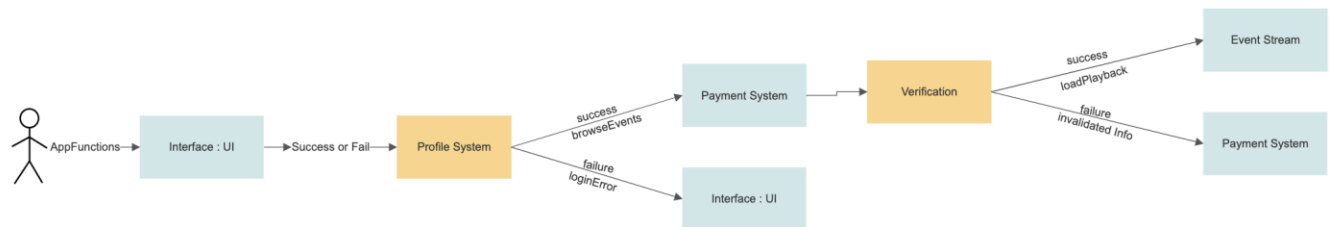
31.3 Actor Identification

There are two actors for the application: registered members and nonmembers. Only members can access browsing and stream events, so nonmembers must create an account and be added to the database before use. Members can browse through or search for events, purchase access, and stream them live or on-demand. Streaming is connected to PlayStation VR Kit. Membership is unrelated to the PlayStation account data, so nonmembers that have PlayStations must create an independent ThriveStream account upon first launch.

31.4 Design Rationale

1.4.1 Architectural Style

The architectural style will be Event-Driven, each time the user interacts with a feature it triggers a change in state, or an event. When a member purchases access to a concert, this is an event, which changes the state of the concert's access, from inaccessible to accessible. When the user streams the concert, another event occurs, transitioning the state of the concert from upcoming to in progress, so the user can begin streaming. Each interaction the user has with the system triggers an event that toggles between states, and each functional requirement of the system has two distinct states.

1.4.2 Design Pattern(s)

The most applicable design patten would be façade because it would provide a simple, unified interface that would be easy for the user to navigate, while ensuring functionality with more complex underlying code structures. Façade pattern would also be helpful in managing the layers of accessibility of functions for the user, such as purchased (access) and not purchased (inaccessible) concerts.

1.4.3 Framework

Because it is crucial to implementation that the application is compatible with both PlayStation and VR technology, ThriveStream would employ the PlayStation VR Dev kits to ensure compatibility between the platform, PlayStation software, and VR hardware. The PlayStation VR Dev kit is only available to developers given access through Sony, so use must be approved first.

32. Functional Design

32.1   Login
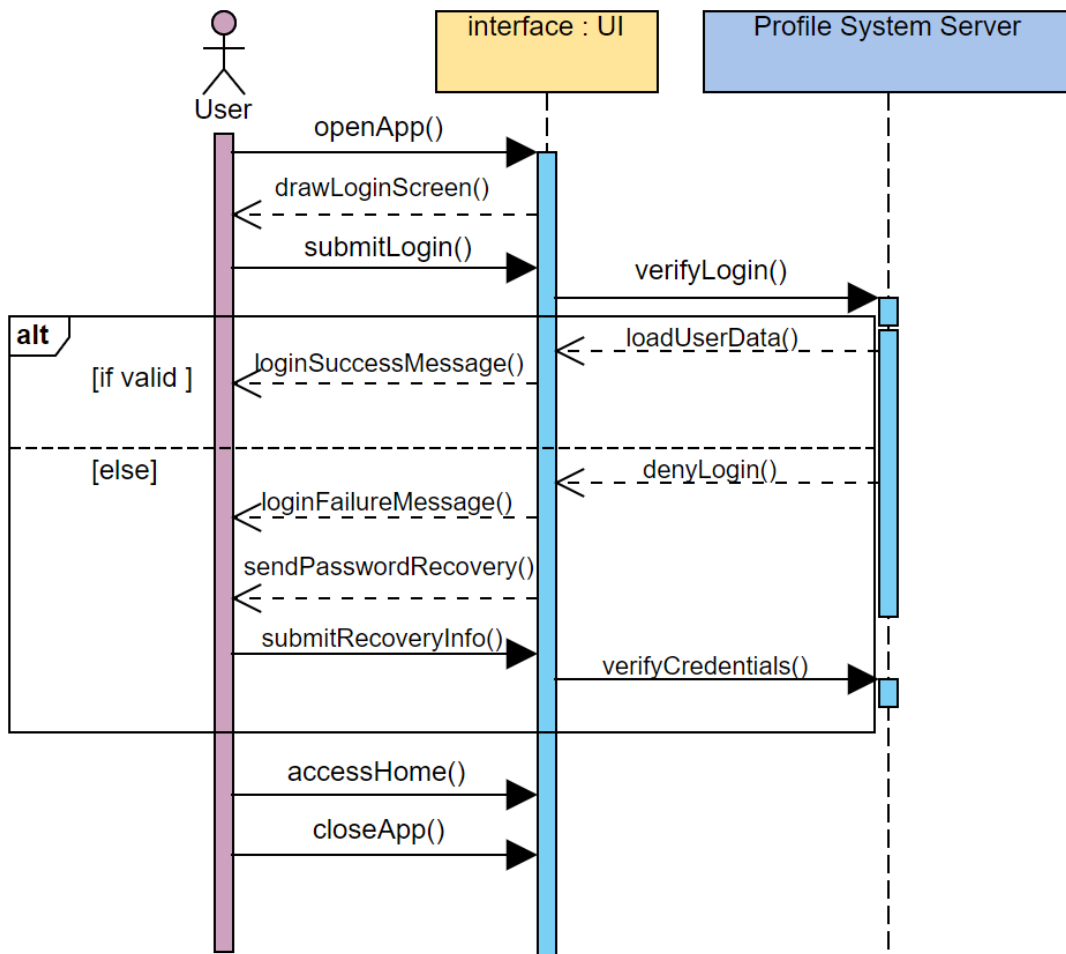
- When the user opens the application the login screen opens
- After submitting the login, the login info is verified
  - o   If the login was successful the user data is loaded and indicated login success
  - o   If the login was unsuccessful the user is notified and can opt for password recovery
- The User can access the homepage or close the app once login is complete

2.2 Sign Up and Edit Account
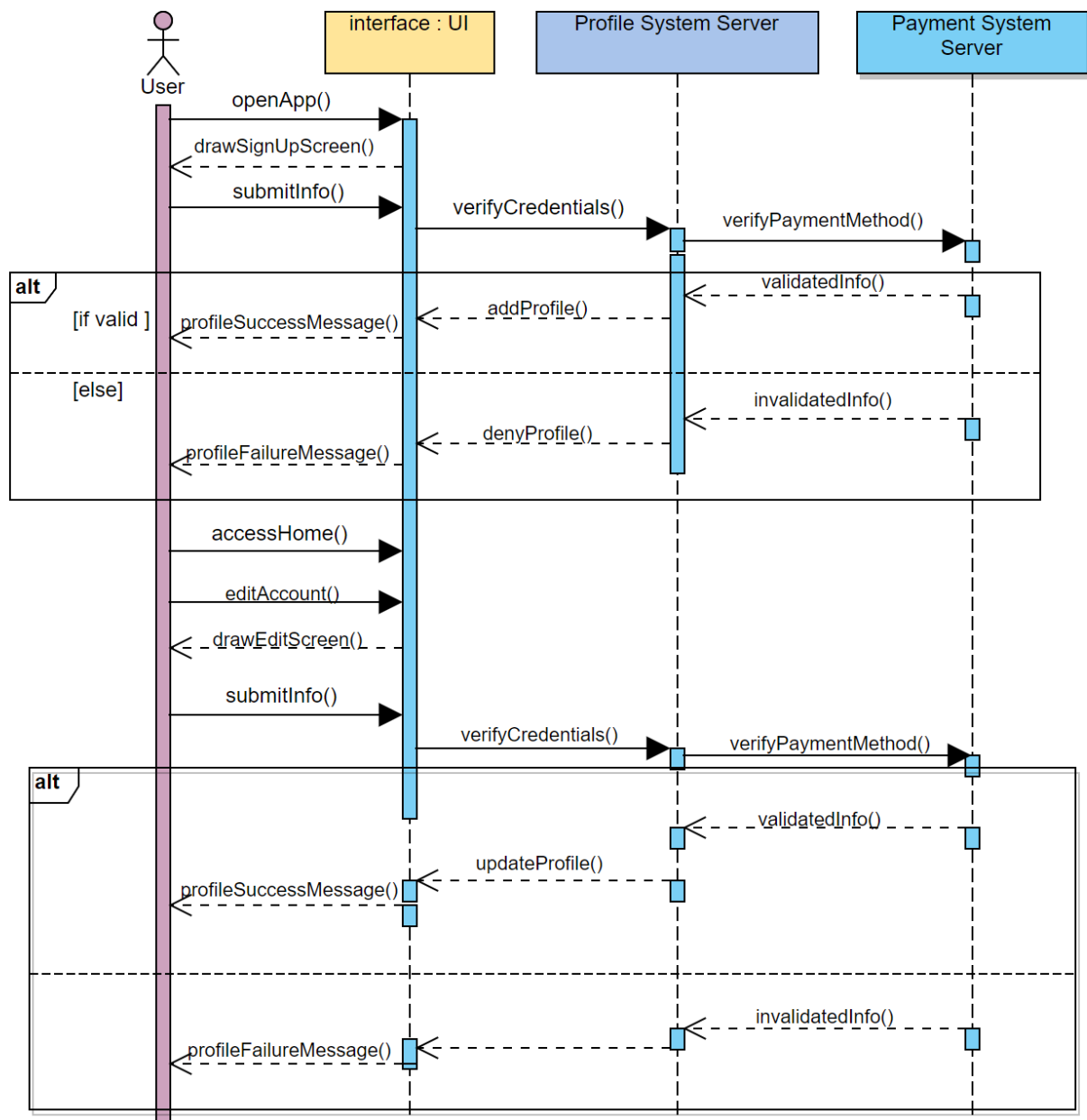
- First time users can create a login using their PlayStation Profile with basic info and payment credentials
- After submitting, the information is verified
  - o If successful, the profile is created and the user is indicated of profile success
  - o If unsuccessful, the profile is denied and the user is indicated of profile failure
- The user can edit their account information and submit new information to be verified
  - o If successful, the profile is updated the user is indicated of profile success
  - o If unsuccessful the user is indicated of profile failure

2.3 Browsing, Searching, Purchasing
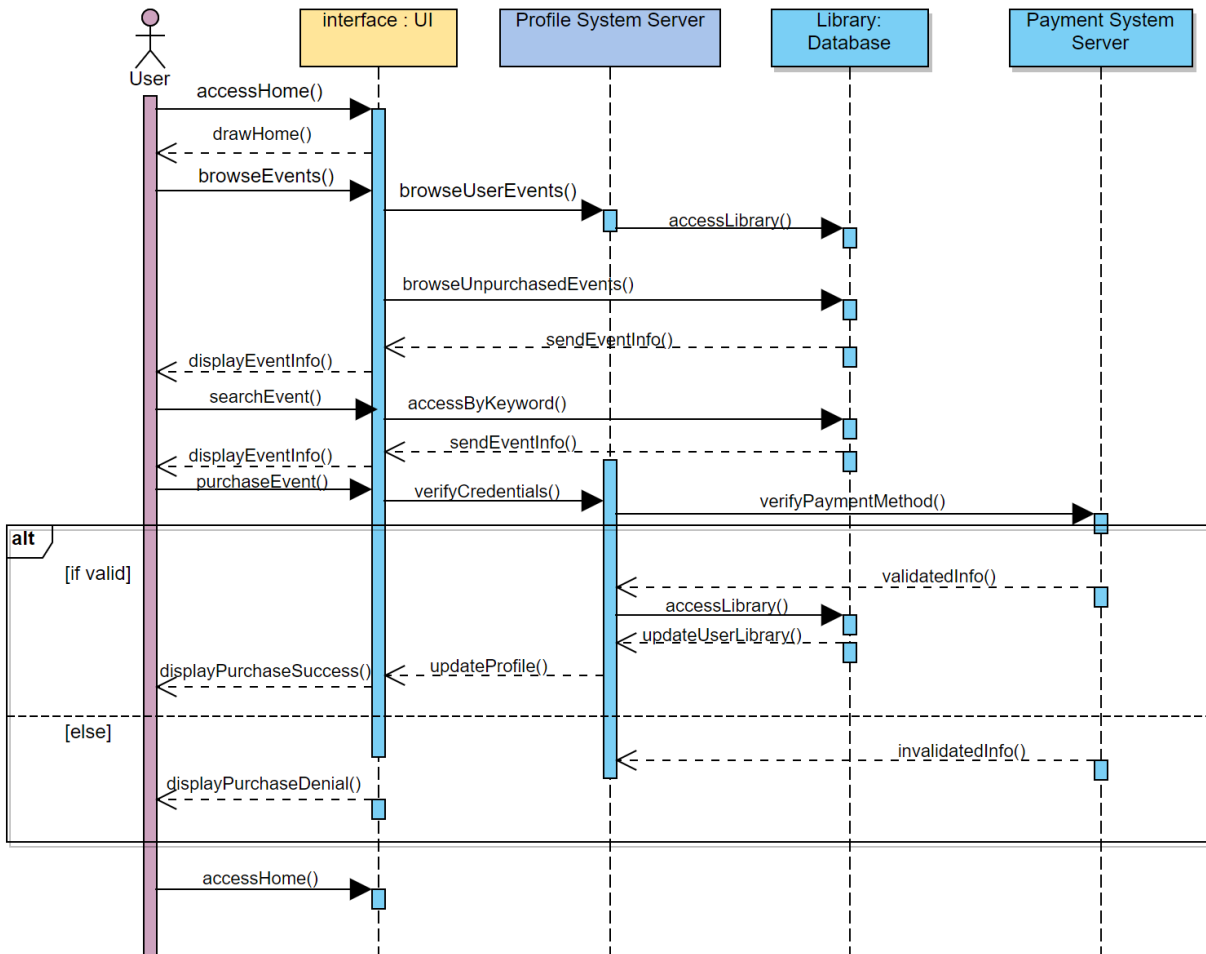
- The logged in user can access home to browse events that are purchased and unpurchased
  - Purchased events are attached to the user's profile/ user library and accessed through the library database
  - Unpurchased events are accessed through the library database
- Event information is sent and displayed to the user
- The user can search for events in the database through keywords and relevant events and their details are returned for viewing
- The user can opt to purchase an event which will start credential verification
  - If the payment method is valid the profile will gain updated access to the event and the user is notified of purchase acceptance
  - If the payment method is invalid the user is notified of denial of purchase
- The user is redirected back to home

2.4 Streaming and Exit
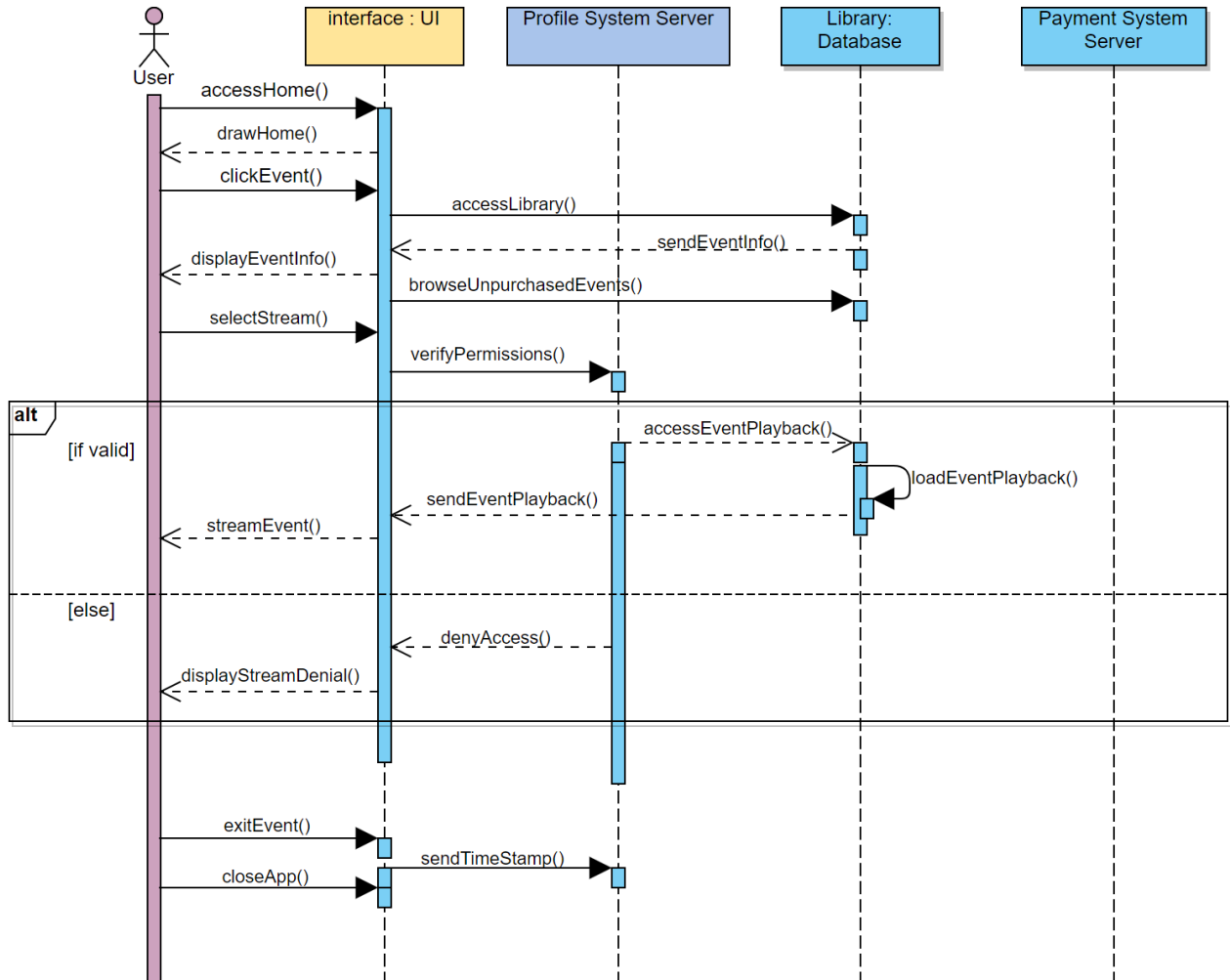
- A logged in user in the homepage can click an event to view the event information and select to stream the event
- In order to stream, the profile has to verify the permissions for streaming access
  - If successful, the profile is granted access to the playback in the library, the playback is recursively loaded as the user is streaming, and the user can view the event
  - If unsuccessful, the user is denied access to the event and is indicated of denial of streaming
- The user can exit the event at any time and the time stamp is sent to the profile to update the playback
- The user can exit the application

33. 3. Structural Design

Class diagram for ThriveStream app to show its components.

Package

SuccessOrFail
- profileSuccessMessage()
- profileFailureMessage()
- displayPurchaseSuccess()
- loginSuccessMessage()
- loginFailureMessage()
- displayPurchaseDenial()
- displayStreamDenial()

AppFunctions
- openApp()
- accessHome(RegisteredUser)
- closeApp()

ScreenDraw
- drawLoginScreen()
- drawSignUpScreen()
- drawEditScreen()
- drawHomeScreen()
- displayEventInfo(Event)

Verification
- verifyCredentials(RegisteredUser): Boolean
- verifyPaymentMethod(): Boolean
- denyAccess()
- accessLibray(RegisteredUser)
- accessEventPlayback(Event)
- validatedInfo()
- invalidatedInfo()
- addProfile(User)
- denyProfile()

« Interface »
BrowseEvents
- browseEvents(): MusicLibrary[*]
- browseUserEvents(): UserLibrary[*]
- searchEvent(): Event[*]
- purchaseEvent(Event)
- clickEvent()
- searchEvent()
- accessByKeyword(String)

Event
- Name: String
- Artist: String
- Date: String
- Purchased: Boolean
- TimeStamp: Integer
- loadEventPlayback(Integer)
- selectStream()
- sendEventPlayback()
- exitEvent()
- sendTimeStamp()

RegisteredUser
- getUserPurchases(): UserLibrary
- submitInfo()
- sendPasswordRecovery()
- submitRecoveryInfo(String)
- editAccount()
- updateProfile()
- loadUserData()
- setUserPurchases()

browse

+getEvent   *

UserLibrary
- Purchased: Event[1..*]
- Unpurchased: Event[1..*]

1   +getUserLibrary

+getUser   1

MusicLibrary
- Library: Event[1..*]
- addEvent(Event)

User
- Name: String
- Email: String
- Registered: Boolean
- getName(): String
- setName(String)
- getPayment(): Integer
- setPayment(Integer)