**Verification Test Plan:**
This section lays out test plans for verification. In addition to the test itself, you will need to clearly explain your test. Be as detailed as possible, identifying what features of your design you are testing, what the test sets/vectors are, and how your selected test(s) cover the targeted feature(s). (Hint: use or modify your design diagram to indicate the target and scope of the tests, what kind of failures you are covering, etc.). Feel free to use any methods discussed in class.

Test cases have been outlined to follow the general chronology of using the app, in which various circumstances are being covered within the breadth of the scenarios. Beginning with login, viewing movie information, purchasing tickets, and the post-viewing instances that occur through the app, the test cases cover the significant features of the app, denoted by the high occurrence of P0 and P1 priority cases. Our team of testers will consist of:

1. Kalanikekai Tran
2. Elijah Agustin
3. Isaac Reveles
4. Jonathan Van

We will be adhering to both black and white box testing. The test cases where we are inputting data and analyzing the output will conform to black box testing, as we are only monitoring the expected outcome and not how the function operated, while for all other test cases we can utilize white box testing as we will want to distinguish what within that test cases' function is not performing as it should. We will test for branch, condition, and statement coverage, as our functions involve for loops, if statements, and bool statements.

I. **Check user login with valid data** (Black box)
    A. Partition testing with valid user login details. Testing that when a user logins with valid information, they will be able to access their account. A valid password and username are required for this test. Testing with valid info will confirm whether users are able to access their accounts, and that the app has saved and logged this user's login information. Username cannot contain any spaces or special characters. Password must be at least 12 characters long, and contain an uppercase letter, a lowercase letter, a number, and a special character, ie. ~`!@#$%^&*()_-+={[}]|\:;'"<,>.?/
        1. Ex. Username: jvan0049
        2. Ex. Password: c0m3T0class!
    B. Variables

        1. userName: String
        2. Password: String

II. **Check user login with invalid data** (Black box)
    A. Partition testing with invalid user login details. Testing that when a user logins with invalid information, they will not be able to access a user account. An invalid password and username are required for this test. Testing with invalid info will confirm that invalid login information will not yield access to any accounts, and that the app has not saved and logged this login information as authenticated credentials. Username cannot contain any spaces or special characters. Password must be at least 12 characters long, and contain an uppercase letter, a lowercase letter, a number, and a special character, ie. ~`!@#$%^&*()_-+={[}]|\:;"'<,>.?/
        1. Ex. Username: kalani tran
        2. Ex. Password: password11
    B. Variables
        1. userName: String
        2. Password: String

III. **Check employee login with valid credentials** (Black box)
    A. Partition testing with valid employee login details. Testing that when an employee logins with valid information, they will be able to access their employee account. A valid password and username are required for this test. Testing with valid info will confirm that invalid info will not yield access to any employee accounts, and that the app has not saved and logged this employee's login information as authenticated credentials. Username is their employee or manager ID. Password must be at least 12 characters long, and contain an uppercase letter, a lowercase letter, a number, and a special character, ie. ~`!@#$%^&*()_-+={[}]|\:;"'<,>.?/
        1. Ex. Username: 131026347
        2. Ex. Password: Eye<3m0V13$
    B. Variables
        1. employeeID: int
        2. managerID: int
        3. Password: String

IV. **Check employee login with invalid credentials** (Black box)
    A. Partition testing with invalid employee login details. Testing that when an employee logins with invalid information, they will not be able to access any employee accounts. An invalid password and username are required

for this test. Testing with invalid info will confirm whether employees are able to access their accounts, and that the app has saved and logged this employee's login information. Username is their employee or manager ID. Password must be at least 12 characters long, and contain an uppercase letter, a lowercase letter, a number, and a special character, ie. ~`!@#$%^&*()_-+={[}]|\:;"'<,>.?/

1. Ex. Username: moe lester
2. Ex. Password: 123456789

B. Variables
   1. employeeID: int
   2. managerID: int
   3. Password: String

V. **Prompt for password reset for user accounts** (Black box)

A. Testing the process of user selecting "Forgot Password?", in which app will prompt users to input their account username or employee ID registered with their account, in which an email will be sent to them with a link to where they will be redirected to enter in a new acceptable password. The tester selecting the "Forgot Password?" option is requisite when assessing this facet. Testing whether the reset password option is viable is essential to ensuring users are still able to access their accounts in the scenario they forget their passwords, have their passwords compromised, etc. Password must be at least 12 characters long, and contain an uppercase letter, a lowercase letter, a number, and a special character, ie. ~`!@#$%^&*()_-+={[}]|\:;"'<,>.?/

B. Variables
   1. Password: String
   2. email: String

VI. **Movie time and date are displayed on the app** (White box)

A. Testing whether users can view the movie times and dates of the currently showing and prospective movies while on the app. This feature is essential so that users are aware of and can browse the available movie showtimes to purchase tickets for. Not having the entire catalog will result in dissatisfied customers; wherein they will miss out on movies they want to watch, or not even have the option to watch a certain movie if it's not included. App must display both dates and showtimes of movies at select theaters.

B. Variables
   1. movieName: String

        2. showingDate: int
        3. showingTime: int
        4. screen: int
        5. location: String
    C. Functions
        1. setName(String): void
        2. getName(): String
        3. setTime(int): void
        4. getTime(): double
        5. setDate(String): void
        6. getDate(): String
        7. setScreen(int): void
        8. getScreen(): int
        9. getLocation(Movie.getName, Movie.getName): String

VII. **Movie theater seat maps are displayed on the app** (White box)

A. Testing whether users can view the seat maps of the currently showing and prospective movies while on the app, including both available and unavailable seats. This feature is essential so that users are aware of and can browse the available seats to purchase tickets for. Users will also be able to view the tier of the seats, such as open-seating vs. premium seating. Not having the entire catalog will result in dissatisfied customers; wherein they will miss out on seats they want to watch from.

B. Variables
1. seatStatus: String
2. seatLocation: <char, int>
3. seatMap: (char, int)

C. Functions
1. getSeatStatus(Movie.getName, Movie.getLocation, Movie.getScreen): String
2. getSeatLocation(getSeatStatus.Movie): <char, int>
3. seatOccupancy(Movie.seatMap()): int

VIII. **IMDb/Rotten Tomatoes reviews are viewable when viewing movies** (White box)

A. Testing whether the IMDb and Rotten Tomatoes scores and reviews are displayed on the app when viewing movies. Must reference the IMDbRT class which holds the databases for both movie critique sites. Testing is essential as including movie reviews and ratings makes for a comprehensive ticket purchasing app, wherein users can make more informed decisions if they are able to view extraneous information.

B. Variables:
   1. IMDb Database
   2. RottenTomatoes Database

C. Functions:
   1. rating: String
   2. getRating(Movie.getName): String
   3. movieName: string
   4. movieRating: int
   5. actors: String

IX. **Successful PearPay transaction with valid payment info** (Black box)

A. Testing whether users can make a successful PearPay transaction when purchasing a movie showtime on the app. This process is essential to make sure that movie transactions can utilize PearPay. Not being able to use PearPay will obstruct any transactions made on the app and will not process any orders made. The app must be able to utilize PearPay to make any type of transaction.

B. Variables
   1. transactionID: String
   2. userTicket: Ticket
   3. validCardInfo: bool

C. Functions:
   1. purchaseNumber(userTicket): int
   2. maxTicket(): bool

X. **Successful implementation of movie review survey the day after the showing** (White box)

A. Testing how the app displays notification to users about movie reviews. This is essential to the app because it gives users the ability to review the movie they watched with functional processes such as commented reviews as well as numbered reviews. Testing is required in order to enable seamless notifications and to ensure the functionality of the review page.

B. Variables:
   1. Movie_review : String
   2. Survey_review : String

C. Functions:
   1. getRating(Movie.getName): String

2. movieName: string
3. movieRating: int