# Camera Rental Application

## Project Agenda: Build a peer-to-peer camera rental application.

## Algorithm

1. Start the program.

2. Initialize the camera list and wallet balance.

3. Display the welcome message and ask the user to login.

4. Read the username and password from the user.

5. If the username and password match the admin credentials, display the login successful message and proceed to the main menu.

6. If the authentication fails, display the authentication failed message.

7. Display the thank you message for visiting the application.

8. Implement the showMainMenu function:

a. Display the main menu options.

b. Read the user's choice.

c. Based on the choice, perform the following actions:

▪ If the choice is 1, call the showMyCameraMenu function.

▪ If the choice is 2, call the rentCamera function.

▪ If the choice is 3, call the viewAllCameras function.

▪ If the choice is 4, call the viewWalletBalance function.

▪ If the choice is 5, exit the program.

▪ If the choice is invalid, display an error message.

9. Implement the showMyCameraMenu function:

a. Display the my camera menu options.

b. Read the user's choice.

c. Based on the choice, perform the following actions:

▪ If the choice is 1, call the addCamera function.

▪ If the choice is 2, call the removeCamera function.

▪ If the choice is 3, call the viewMyCameras function.

▪ If the choice is 4, return to the previous menu.

▪ If the choice is invalid, display an error message.

10.Implement the addCamera function:

a. Generate a new cameraId by incrementing the size of the camera list.

b. Read the camera brand, model, and price from the user.

c. Set the camera availability to true.

d. Create a new cameraData object with the provided information.

e. Add the camera to the camera list.

f. Display a success message.

g. Call the viewAllCameras function.

11.Implement the removeCamera function:

a. Read the cameraId from the user.

b. Iterate through the camera list.

c. If a camera with a matching cameraId is found:

▪ Remove the camera from the list.

▪ Display a success message.

▪ Set the "removed" flag to true.

▪ Break out of the loop.

d. If the "removed" flag is false, display a not found message.

e. Call the viewAllCameras function.

12.Implement the viewMyCameras function:

a. Display a table header.

b. Iterate through the camera list.

c. For each camera, display the camera details.

d. Display a table footer.

13.Implement the rentCamera function:

a. Display a table header.

b. Iterate through the camera list.

c. For each available camera, display the camera details.

d. Display a table footer.

e. Read the cameraId from the user.

f. Iterate through the camera list.

g. If a camera with a matching cameraId is found and the wallet balance is sufficient:

▪ Set the camera availability to false.

▪ Subtract the camera price from the wallet balance.

▪ Display a success message.

h. If the camera is not found or the wallet balance is insufficient, display an appropriate error message.

14.Implement the viewAllCameras function:

a. Display a table header.

b. Iterate through the camera list.

c. For each camera, display the camera details.

d. Display a table footer.

15.Implement the viewWalletBalance function:

a. Display the current wallet balance.

b. Read the user's choice to deposit more amount.

c. If the choice is 1, read the deposit amount and add it to the wallet balance.

d. Display the updated wallet balance.

16.End the program.

**Number and Duration of Sprints Required**

o Sprint 1:

Duration: 1 working day

o Sprint Goal: User Authentication and Camera Listing

o User stories/tasks:

▪ Implement the login functionality.

▪ Create the cameraData class.

▪ Initialize the camera list and wallet balance.

▪ Implement the viewAllCameras function.

▪ Implement the login and showMainMenu functions.

o Sprint 2:

Duration: 1 working day

o Sprint Goal: Camera Rental and My Camera Management

o User stories/tasks:

▪ Implement the rentCamera function.

▪ Implement the viewMyCameras function.

▪ Implement the addCamera and removeCamera functions.

▪ Add error handling and validation to the existing code.

▪ Test and debug the implemented features.

o Sprint 3:

Duration: 1 working day

o Sprint Goal: Wallet Balance and Final Testing

o User stories/tasks:

▪ Implement the viewWalletBalance function.

▪ Test and debug the entire application.

▪ Refactor code for better readability and maintainability.

▪ Perform code reviews and address any issues.

▪ Conduct comprehensive testing to ensure functionality and correctness