DUGBET USE-CASE SPECIFICATION

Group 03

Huynh Huu Phuc

Tu Canh Minh

Nguyen Truc Nhu Binh

Nguyen Duc Hung

Hoang Nhu Vinh

Version 1.12

Revision History

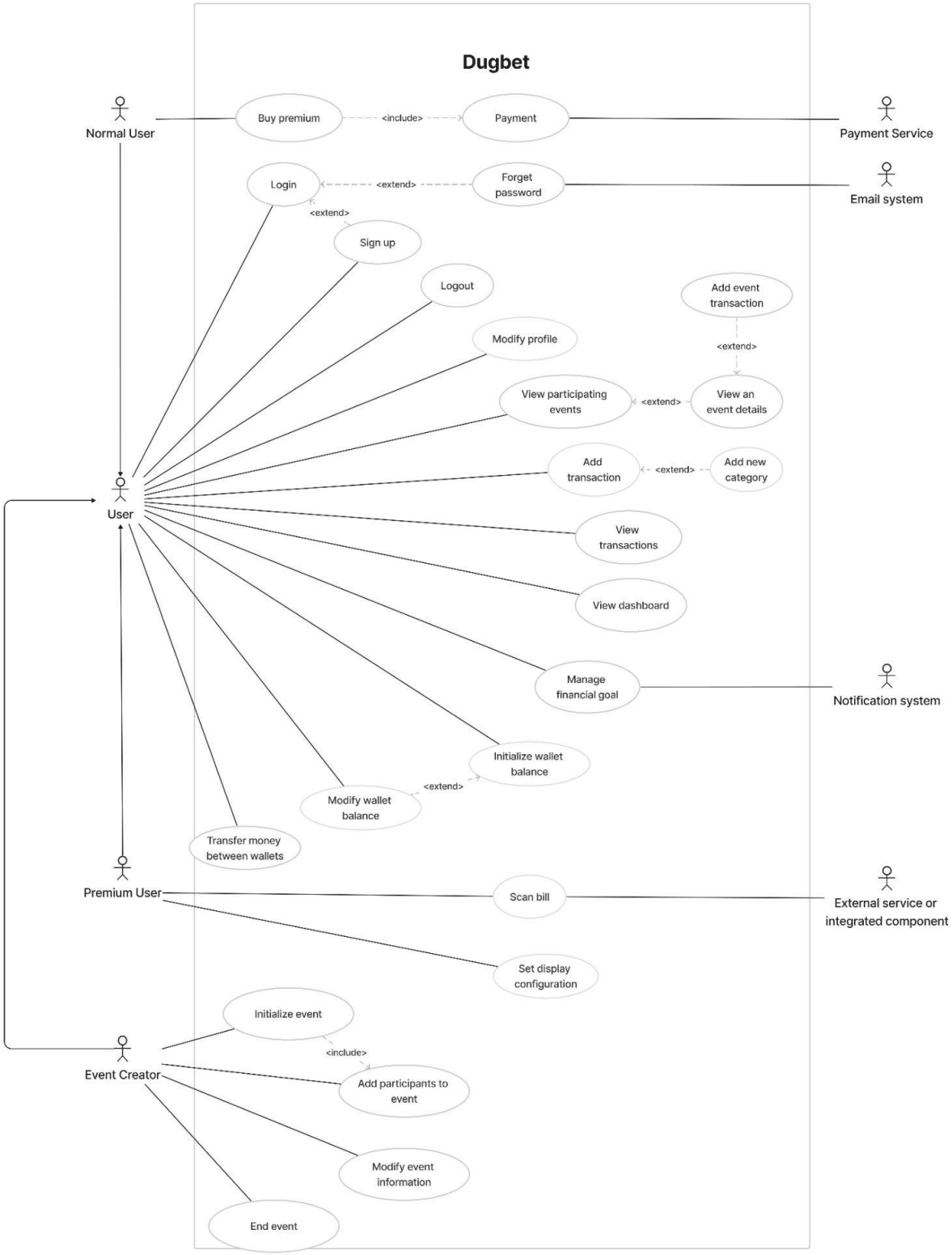
Date	Version	Description	Author
08/11/2023	1.0	Initial version, Reformat Change headers' content	Nguyen Truc Nhu Binh
14/11/2023	1.1	Append login, sign up and forget password use case specifications	Huynh Huu Phuc
15/11/2023	1.2	Append Initialize wallet balance, Wallet transfer, Modify profile, Set display configuration	Nguyen Duc Hung
15/11/2023	1.3	Add use-case specifications for features related to Event finance tracking	Tu Canh Minh
17/11/2023	1.4	Remove use-case Quit event and add use-case End event	Tu Canh Minh
17/11/2023	1.5	Add use-case diagram	Nguyen Duc Hung
17/11/2023	1.6	Add use-case specifications for features related to Add transaction	Hoang Nhu Vinh
18/11/2023	1.7	Redraw use case diagram	Huynh Huu Phuc
18/11/2023	1.8	Add use-case specifications for features related to Scan bill	Hoang Nhu Vinh
18/11/2023	1.9	Add use-case specifications for Modify Wallet Balance	Nguyen Duc Hung
18/11/2023	1.10	Append buy premium, make payment, add a new category, view transaction, view statistic, manage financial goal use case specification Edit use case diagram	Huynh Huu Phuc
18/11/2023	1.11	Update and modify use-case specifications for features related to View transaction, view statistic. Review use-case specifications and update some missing flow, incorrect information. Make use-case specifications more details.	Hoang Nhu Vinh
18/11/2023	1.12	Review all use-case specifications and check the consistency between them	Nguyen Truc Nhu Binh

		and use-case specifications. Fix grammar mistakes	

Table of Contents

Use-Case Specification

1. USE-CASE DIAGRAM


Dugbet
Normal User
Buy premium
<include>
Payment
Payment Service
Login
<extend>
Forget password
Email system
<extend>
Sign up
Logout
Add event transaction
<extend>
Modify profile
View participating events
<extend>
View an event details
Add transaction
<extend>
Add new category
User
View transactions
View dashboard
Manage financial goal
Notification system
Initialize wallet balance
<extend>
Modify wallet balance
Transfer money between wallets
Premium User
Scan bill
External service or integrated component
Set display configuration
Initialize event
<include>
Event Creator
Add participants to event
Modify event information
End event

2. USE-CASE SPECIFICATIONS

2.1 Use-case: Login

Use Case Name	Login
Brief description	This use case describes how the User can log in to the Dugbet application
Actor(s)	User
Basic flow	1. User opens the Dugbet application 2. System displays the Sign in screen 3. User enters the username and password to corresponding field on the screen 4. User clicks 'Get Started' button on the screen 5. System validates the user's input to ensure it is correct. 6. System authenticates the account with data stored in the database. 7. System navigate to the Home screen or Admin screen depend on the role of account
Alternative flow	**Alternative flow 1: Login information is invalid** 1. At step #5 of basic flow, the system displays an error message about the invalid input fields. 2. The use case continues step #2 in the basic flow **Alternative flow 2: Login authentication failed** 1. At step #6 of basic flow, the system displays a message alerting the wrong username or password 2. Use case continue step #2 in the basic flow
Pre-conditions	• User account has been created • User's device connected to the internet
Post-conditions	• User successfully logs into account.
Non-Functional Requirement	• The password is hashed using MD5

2.2 Use-case: Sign up

Use Case Name	Sign up
Brief description	This use case describes how the User can register a new Dugbet account.
Actor(s)	User
Basic flow	1. User clicks the 'Create Account' button on screen 2. System displays the Signup screen 3. User enters the email, username, and password to the corresponding field on the screen

	4. User clicks the 'Sign Up' button on screen 5. System validates the user's input to ensure it is correct. 6. System creates a new token for the account in the database. 7. System navigate to the Home screen
Alternative flow	**Alternative flow 1: User already has account** 1. At step #1 of basic flow, the user clicks the 'Already have an Account' button on screen 2. The system navigates to the sign-in screen and invokes the use case Login **Alternative flow 2: (#5). Email is not valid** 1. The system displays an error message about the invalid email fields. 2. The use case continues step #3 in the basic flow **Alternative flow 3: (#5). Username is used** 1. The system displays an error message about the invalid username field. 2. The use case continues step #3 in the basic flow **Alternative flow 4: (#5) The passwords is not valid** 1. The system displays an error message about the invalid password field (e.g. must have at least 8 characters). 2. The use case continues step #3 in the basic flow
Pre-conditions	● User's device connected to the internet
Post-conditions	● User successfully creates a new account.
Non-Functional Requirement	● The password is hashed using MD5

2.3 Use-case: Forget password

Use Case Name	Forget password
Brief description	Allow user to reset account password via email
Actor(s)	User
Basic flow	1. User clicks 'Forget password?' button on the screen 2. System displays the Forget password screen 3. User entry the email that link with account to corresponding field on the screen 4. User clicks 'Send Email' button on the screen 5. System validates the user's input to ensure email is valid. 6. System sends a reset password link instruction to the email. 7. System alert message that reset instruction has been send to email 8. User clicks 'Back to Sign in' button on the screen
Alternative flow	**Alternative flow 1: (#7). Resend reset password link** 1. User clicks the 'Resend email again' button on the screen 2. The use case continue step #7 in the basic flow

Pre-conditions	• User in the Sign in screen • User's device connected to the internet
Post-conditions	• User successfully reset his/her password…
Non-Functional Requirement	• The password is hashed using MD5 • The waiting time for reset password link is due after 60 seconds

2.4 Use-case: Logout

Use Case Name	Logout
Brief description	Allow user to log out account of Dugbet application
Actor(s)	User
Basic flow	1. User clicks the 'Logout' button on the screen
Alternative flow	None
Pre-conditions	• User in the Setting screen
Post-conditions	• User logs out account of Dugbet application.
Non-Functional Requirement	None

2.5 Use-case: Initialize Wallet Balance

Use-case name	Initialize Wallet Balance
Brief description	The use case describes how the user can create the wallet balance and initialize wallet balance.
Actor(s)	User

Basic flow	1. At Home screen, the user clicks the wallet icon at the bottom navigation bar to navigate to the wallet management page. 2. The system displays the wallet management page. 3. User clicks the personal wallet tab on the wallet management page. 4. The system displays the personal wallet. 5. User clicks on “plus” at the floating button to create the wallet. 6. The system displays a form for the user to input information about the new wallet. 7. User inputs the information and wallet balance. 8. User clicks on the “Create” button. 9. The system examines the conditions. 10. The system initializes the wallet with the specified amount. 11. The system navigates to the wallet management page. 12. The system sends a message to the user “The wallet balance has been initialized”.
Alternative flow	**Alternative flow 1: User does not fill in all the information of the wallet.** 1. After #8 in the basic flow, the system sends a message to the user “Please fill in all the information”. 2. User fills in the empty information 3. Continue step #8 in the basic flow. **Alternative flow 2: The wallet name existed.** 1. After #9 in the basic flow, the system sends a message to the user “Wallet existed.”. 2. The system displays existing wallet information along with the entered amount of money. 3. Continue step #8 in the basic flow of the use-case Modify Wallet Balance.
Pre-conditions	User has installed and logged into the Home screen of the Dugbet application successfully
Post-conditions	User's wallet balance has been successfully created with the specified amount.

Non-Functional Requirements	• The system has the ability to process initialization after 2s. • The system must be able to handle invalid user input correctly.

2.6 Use-case: Modify Wallet Balance

Use-case name	Modify Wallet Balance
Brief description	The use case describes how the user can change the wallet balance of the specified wallet
Actor(s)	User
Basic flow	1. At the Home screen, the user clicks the wallet icon at the bottom navigation bar to navigate to the Wallet page. 2. The system displays the Wallet page. 3. User clicks the personal wallet tab on the Wallet page. 4. The system displays the list of personal wallets. 5. User clicks on the wallet which is needed to change the balance. 6. The system displays a form for the user with detailed information about the wallet. 7. User clicks on the pen icon next to the balance of the form. 8. User inputs the wallet balance. 9. User clicks on the “Save” button. 10. The system examines the conditions. 11. The system saves the wallet with the specified amount. 12. The system navigates to the wallet management page. 13. The system sends a message to the user “The wallet balance has been changed”.

Alternative flow	**Alternative flow 1: User inputs the invalid balance in the wallet (smaller than 0, special characters, etc.).** 4. After #10 in the basic flow, the system sends a message to the user "Your input is invalid. Please check again". 5. The system displays the form containing the information just entered. 6. Continue step #7 in the basic flow. **Alternative flow 2: The user wants to stop the modification process.** 1. At #7, #8, or #9 in the basic flow, the user clicks on the right-arrow icon. 2. The system saves the initial inputs from the user on the clipboards. 3. The system navigates to the wallet management page.
Pre-conditions	User has installed and logged into the Home screen of Dugbet application successfully.
Post-conditions	User's wallet balance has been successfully changed to the specified amount.
Non-Functional Requirements	● The system has the ability to process initialization after 2s. ● The system must be able to handle invalid user input correctly.

2.7 Use-case: Transfer money between wallets

Use-case name	Transfer money between wallets
Brief description	Wallet Transfer describes how the user can transfer money from wallets to other users in the system.
Actor(s)	User

Basic flow	1. At Home screen, user clicks the wallet icon at the bottom navigation bar to navigate to the wallet page. 2. The system displays the wallet page. 3. User clicks on the Personal tab. 4. The system displays a list of wallets. 5. User clicks on the "..." icon at the right-top of the wallet which is needed to transfer. 6. The system displays options for this wallet. 7. User clicks on the transfer option. 8. The system displays the form to input amount of money, content of transaction, and wallet receiving amount of money. 9. User fills in and chooses all the information. 10. User clicks on "Transfer" button at the end of the form. 11. The system examines the conditions. 12. The system performs the request. 13. The system navigates to the wallet page. 14. The system sends a message to the user "The transfer is successful".
Alternative flow	**Alternative flow 1: User does not fill in all the information** 1. After #9 in the basic flow, the system sends a message to the user "Please fill in and choose all the information". 2. User fills in and chooses the empty input. 3. Continue step #10 in the basic flow. **Alternative flow 2: The wallet does not have sufficient money.** 1. After step #10 in the basic flow, the system navigates to the wallet page. 2. The system sends a message to the user "Balance of wallet does not have enough money".
Pre-conditions	● User has installed and logged into the Home screen of Dugbet application successfully ● User has a sufficient balance in their wallet to cover the transfer amount.
Post-conditions	The transfer amount is reduced from the user's wallet balance and added to the account of specified wallet.

Non-Functional Requirements	• The system has the ability to process wallet transfers after 3s. • The system must be able to handle invalid user input correctly.

2.8 Use-case: Modify Profile

Use-case name	Modify Profile
Brief description	The use case describes how the user can change the information in his/her profile. (Ex: name, email, etc.)
Actor(s)	User
Basic flow	1. At the Home screen, user clicks on the “Welcome” button in the navbar to navigate to the profile modification page. 2. The system displays the profile modification page. 3. User inputs the desired changes to his/her profile information. 4. User clicks the "Save" button. 5. The system examines the user's input. 6. The system updates the user's profile information. 7. The system navigates to the Home screen. 8. The system sends a confirmation message to the user “Profile has been updated”.

Alternative flow	**Alternative flow 1: The user wants to stop the modification process.** 1. After steps #2 or #3 in the basic flow, the user clicks on the right-arrow to exit. 2. The system saves the initial inputs from user. 3. The system navigates to the Home screen. **Alternative flow 2: The user changes the information of the profile with invalid format.** 1. After step #5 in the basic flow, the system sends a message to the user “Please fix the input with right format”. 2. The user fixes the input in the right format. 3. Continue with step #4 in the basic flow. **Alternative flow 3: The user does not perform any change.** 1. After step #5 in the basic flow, the system sends a message to the user “There are no changes in the user’s inputs”. 2. Continue with step #3 in the basic flow. **Alternative flow 4: Access via Setting screen.** 1. At the Home screen, user clicks the “Setting” icon at the bottom navigation bar to navigate to the Setting screen. 2. The system displays the Setting screen. 3. User user clicks on the Profile options on the Setting screen. 4. Continue step #2 in the basic flow.
Pre-conditions	User has installed and logged into the Home screen of Dugbet application successfully
Post-conditions	User's profile information is updated with the specified changes.
Non-Functional Requirements	● The system must be able to process modification changes after 2s. ● The system must be secure and protect user information. ● The system must be able to handle invalid user input correctly.

2.9 Use-case: Set Display Configuration

Use-case name	Set display configuration
Brief description	The use case describes how to change the display configuration(Ex: background, font, etc.)
Actor(s)	Premium User
Basic flow	1. At Home screen, user clicks on the setting icon at the bottom navigation bar to navigate to the display configuration page. 2. The system displays the configuration page. 3. User selects the desired display configuration options. 4. User clicks on the “Save” button. 5. The system updates the system's display configuration. 6. The system navigates to the Home screen. 7. The system sends a confirmation message to the user “The display configuration has been updated”.
Alternative flow	**Alternative flow 1: User wants to stop the display configuration process.** 1. From steps #2 or #3 in the basic flow, user clicks on the right-arrow icon. 2. The system saves the initial options from the user. 3. The system navigates to the Home screen. **Alternative flow 2: The display setting does not change.** 1. From step #4 in the basic flow, the system sends a message to the user “Display setting configuration does not change”. 2. Continue with step #3 in the basic flow.
Pre-conditions	User has installed and logged into the Home screen of Dugbet application successfully
Post-conditions	The application's display configuration is updated with the specified changes.

Non-Functional Requirements	The system must be able to process display configuration changes after 2s.

2.10 Use-case: View participating events

Use-case name	View participating events
Brief description	This use-case describes how both a normal user and a premium user can track the events they are joining and how they can manage those events activities.
Actor(s)	User
Basic flow	1. At Home screen, user clicks the wallet icon at the bottom navigation bar to navigate to the wallet page. 2. The system displays the wallet page. 3. User clicks on the Event tab to enter the screen of their participating events. 4. The system navigates to the Event page. 5. The system loads all the events in which the current user is participating from the database. 6. The system displays the loaded events on the screen in the form of a scrollable list.
Alternative flow	**Alternative flow 1: The system fails to load the events from the database** 1. After step #5, the system announces to the user that it has failed to load their participating events. 2. The system displays a button for the user to navigate to the Home page. **Alternative flow 2: The user has not participated in any event** 1. After step #5, the system announces to the user that they are currently not participating in any event. 2. The system displays a button for the user to navigate to the Home page.

Pre-conditions	User has logged into the Home screen of Dugbet application successfully
Post-conditions	The system successfully displays all the events that the current user is participating on the device's screen.
Non-Functional Requirements	The system should be able to display user's events on the screen after 5s.

2.11 Use-case: View event details

Use-case name	View event details
Brief description	This use-case describes how both a normal user and a premium user can manage a specific event that they are participating in. This use-case extends the **View participating events** use-case.
Actor(s)	User
Basic flow	1. On the "Event" page, the user clicks on any specific event to view more details about that event. 2. The system loads information related to the event from the database. 3. The system displays relevant information about the event on the screen, including event name, date of creation, list of members, list of occurred transactions, …
Alternative flow	**Alternative flow 1: The system fails to load details of the event** 3. After step #2, the system announces to the user that it has failed to load details of the event. 4. The system displays a button for the user to navigate to the Event page.

Pre-conditions	User is on the Event page.
Post-conditions	The system successfully displays details of the event on the device's screen.
Non-Functional Requirements	The system should be able to display details of the event on the screen after 5s.

2.12 Use-case: Add event transaction

Use-case name	Add event transaction
Brief description	This use-case describes how a participant in an event can record a new expense of the event by adding a new transaction. This use-case extends the **View event details** use-case.
Actor(s)	User
Basic flow	1. On the "Event details" page, the user clicks on a button that allows them to add a new transaction to the event. 2. The system displays a window for the user to input the transaction information, including the amount of money, category, members involved in this transaction, … 3. The user hits Enter on the keyboard or clicks the Done button to finish recording the expense. 4. The system loads the information of this new expense onto the database.
Alternative flow	**Alternative flow 1: The system fails to load the expense information onto the database** 1. After step #4, the system announces to the user that it has failed to load the expense details to the server. 2. The system displays an option for the user to navigate back to the "Event details" page, and another option for the user to re-enter the details of the transaction.

	Alternative flow 2: The user has not entered all the required information of the transaction 1. After step #3, the system announces that the user has not entered all the required information for the transaction. 2. The system provides two options for the user: a. They can continue to enter the remaining required pieces of information. b. They can cancel the current transaction and return to the "Event details" page.
Pre-conditions	User is on the "Event details" page.
Post-conditions	The system successfully loads the information about the transaction onto the server or aborts the transaction if the user chooses to cancel it.
Non-Functional Requirements	The system should be able to load the transactional information onto the server after 5s.

2.13 Use-case: Initialize event

Use-case name	Initialize event
Brief description	This use-case describes how a user can create an event and add other users to the event as members.
Actor(s)	User
Basic flow	1. On the "Event" page, the premium user can click a button to create a new event. 2. The system displays a window for the user to input basic information related to the new event, e.g. event's name. 3. Initiate use-case **Add participants to event**. 4. The user clicks the Done button to finish creating the event. 5. The system loads the inputted information onto the server. 6. The system navigates to the "Event details" page with data entered from the previous step is displayed on the screen.

Alternative flow	**Alternative flow 1: The system fails to load the event information onto the server** 1. After step #5, the system announces to the user that it has failed to process the request to create a new event. 2. The system provides the user with two options: a. Retry the creation of new event. b. Navigate back to the "Event" screen. **Alternative flow 2: The user has not entered all the required information for the new event** 1. After step #4, the system announces to the user that they have not filled all the required information to create a new event. 2. The system provides the user with two options: a. Enter the missing information. b. Abort the creation and navigate back to the "Event" screen.
Pre-conditions	User is on the "Event" screen and is a premium user.
Post-conditions	The system successfully loads the new event's information onto the server and creates the event or aborts the request if the user chooses to cancel it.
Non-Functional Requirements	The system should be able to load the event's information onto the server and display the new event after 5s.

2.14 Use-case: Add participants to event

Use-case name	Add participants to event
Brief description	This use-case describes how an event creator can add other users to the event being created.
Actor(s)	Event creator

Basic flow	1. On the window displayed from the use-case **Initialize event**, the system displays a text field for the user to enter the username to add to the event being created. 2. The user enters the desired username to be added to the event. When they do so, the system shows a dropdown menu for usernames that match the prefix being inputted by the user. 3. The user picks the username they wish. 4. The system inserts a box representing that username into the text field. 5. If the user does not finish, go to step #2. Otherwise, stop.
Alternative flow	
Pre-conditions	The window from the **Initialize event** use-case is displayed.
Post-conditions	The system successfully loads the usernames being prompted by the user.
Non-Functional Requirements	The system should be able to load the username being prompted from the server and display it on the dropdown menu after 5s.

2.15 Use-case: Modify event information

Use-case name	Modify event information
Brief description	This use-case describes how a premium user can update the information of an event that they have created.
Actor(s)	Event creator

Basic flow	1. On the “Event” screen, the user chooses a specific event to update its information. 2. The system displays the window that contains information related to the event. 3. The user navigates to a specific field to update its information. 4. The user clicks the Done button to finish updating the event’s information. 5. The system loads the updated information about the event onto the server.
Alternative flow	**Alternative flow 1: The system fails to load the event information onto the server** 1. After step #5, the system announces to the user that it has failed to process the request to update the event’s information. 2. The system provides the user with two options: a. Retry the update of the event. b. Navigate back to the “Event” screen. **Alternative flow 2: The premium user has removed information of a required field.** 1. After step #4, the system announces to the user that they have made information about the event incomplete by removing data of a required field. 2. The system provides the user with two options: a. Require the user to fill in the missing compulsory fields. b. Abort the update and navigate back to the “Event” screen.
Pre-conditions	The user is on the “Event” screen.
Post-conditions	The system successfully loads the updated information about the event onto the server or aborts the update if the user takes invalid actions or the system fails to loads the information onto the server (as described in the alternative flows).
Non-Functional Requirements	The system should be able to load the username being prompted from the server and display it on the dropdown menu after 5s.

2.16 Use-case: End event

Use-case name	End event
Brief description	This use-case describes how a user who has created this event can close the event.
Actor(s)	Event creator
Basic flow	1. On the “Event details” page, the event creator clicks on the Close button to close this event. 2. The system displays a popup window to confirm the closure. 3. The user clicks the button to confirm the closing. 4. The system transfers the information to the server. 5. The server updates the database appropriately so that further transactions cannot be added to the closed event.
Alternative flow	**Alternative flow 1: The system fails to load the closing information onto the server** 1. After step #4, the system announces to the user that it has failed to process the request to close the event. 2. The system provides the user with two options: a. Try closing the event again. b. Navigate back to the “Event details” screen. **Alternative flow 2: The event creator cancels closing the event** 1. After step #2, the event creator chooses to cancel the closure. 2. The application navigates back to the “Event details” screen.
Pre-conditions	The user is on the “Event details” screen.
Post-conditions	The system successfully loads the information about the event being closed onto the server or does nothing if the user chooses to cancel the closure.

Non-Functional Requirements	The system should be able to load the information onto the server after at most 5s.

2.17 Use-case: Add transaction

Use-case name	Add Transaction
Brief description	This use-case describes the steps to adding a new financial transaction to the Dugbet app.
Actor(s)	User
Basic flow	1. Users open the Dugbet app and navigate to the "Add Transaction" feature. 2. Enter transaction details: ● Transaction type (income or expense). ● Amount of the transaction. ● Date and time of the transaction. ● Category of the transaction (e.g., food, transportation, salary). ● Personal or event transaction? ● Additional notes (optional). 3. The user reviews the entered information. 4. The user clicks on the "Save" or "Add" button to confirm the transaction. 5. System Validate and display the status of the transaction. 6. System Update account balance, log to history if transaction added successfully. 7. System Provides a confirmation message for users.
Alternative flow	**Alternative flow 1: (after #7) The system fails to load the expense information onto the database** 1. The app displays an error message and prompts the user to correct the errors. **Alternative flow 2: (after #5) The user has not entered all the required information of the transaction** 3. After step #3, the system announces that the user has not entered all the required information for the transaction. 4. The system provides two options for the user:

	a. They can continue to enter the remaining required pieces of information. b. They can cancel the current transaction and return to the Home screen. **Alternative flow 3 (after #4): If the user cancels the transaction** 5. The app discards the transaction details and returns to the main menu. **Alternative flow 4 (after #2): If the user chooses to add the transaction to a particular event** 6. The user inputs further details about the event and the members of that event who are involved in the transaction.
Pre-conditions	User is on the Home screen.
Post-conditions	The system successfully loads the information about the transaction onto the server or aborts the transaction if the user chooses to cancel it.
Non-Functional Requirements	The system should be able to load the transactional information onto the server after 5s.

2.18 Use-case: Scan bill

Use-case name	Scan bill
Brief description	This use-case describes the Bill Scanner feature in the Dubget app, allowing users to scan and automatically input transaction details from bills or receipts into their financial records.
Actor(s)	Premium user, OCR Engine (external service or integrated component)

Basic flow	1. Users open the Dugbet app 2. User navigate to the "Bill Scanner" button. 3. The app activates the device's camera. 4. The user captures an image of the bill or receipt containing transaction details. 5. The app utilizes an OCR engine to extract relevant information from the scanned image, such as: ● Transaction type (income or expense). ● Amount of the transaction. ● Date and time of the transaction. ● Category of the transaction (e.g., food, transportation, salary). ● Additional notes (optional). 6. The app presents the extracted information to the user for review 7. User can edit information if needed. 8. The user clicks on the "Save" or "Add" button to confirm the transaction. 9. System validate and display the status of the transaction. 10. System update account balance, log to history if transaction added successfully. 11. System provides a confirmation message for users.
Alternative flow	**Alternative flow 1: (after #5) If the OCR engine fails to accurately extract information** 1. The app notifies the user of the failure. 2. The user may choose to edit the information manually or retry the scanning process. **Alternative flow 2: (after #10) The system fails to load the expense information onto the database** 1. The app displays an error message and prompts the user to correct the errors. **Alternative flow 3: The user can enter transaction details manually if they prefer not to use Bill Scanner** 1. The app displays if user want to manual enter transaction or not. a. If they want, navigate to “Add transaction” b. Else end the transaction and back to the homepage. **Alternative flow 4: If the user cancels the transaction** 1. The app discards the transaction details and returns to the main

	menu.
Pre-conditions	• The users log in to the premium accounts. • The device has a working camera.
Post-conditions	• The system successfully loads the information about the transaction onto the server or aborts the transaction if the user chooses to cancel it. • The transaction from the scanned bill is added to the user's transaction history.
Non-Functional Requirements	• The system should be able to load the transactional information onto the server after 5s. • The OCR engine should be reliable and accurate in extracting information from various types of bills and receipts in less than 5s.

2.19 Use-case: Buy premium

Use Case Name	Buy premium
Brief description	This use case describes how the normal user pay for upgrade to premium account
Actor(s)	Normal user
Basic flow	1. User clicks 'Premium' button on the screen 2. System displays the information and benefits of premium user on the screen 3. User clicks 'Buy premium now' button on the screen 4. System invokes then use case Make payment 5. System displays the successfully purchase notification
Alternative flow	**Alternative flow 1: Payment failed** 1. The system displays a message that payment has failed. 2. The use case continue step #2 of the basic flow
Pre-conditions	• User in the Setting screen
Post-conditions	• User successfully upgrade account to premium
Non-Functional Requirement	• The reposending time is less than 2s

2.20 Use-case: Make payment

Use Case Name	Make payment
Brief description	This use case allow user to make a payment via payment service
Actor(s)	Normal user, Payment service
Basic flow	1. User selects payment type and entry personal information 2. System displays the total amount of payment on the screen 3. User verify the payment amount 4. User click 'Confirm' button on the screen to process payment 5. System displays an successfully payment notification
Alternative flow	**Alternative flow 1: Change payment type** 1. From step #2 of basis flow, user clicks 'Back' button 2. The use case continue at step #1 of basic flow **Alternative flow 2: Payment fail** 1. After step #4, system displays an unsuccessfully payment notification 2. Use case ends
Pre-conditions	● User account type is currently normal user ● User's device connected to the internet ● User has a payment account
Post-conditions	● User successfully make a purchase
Non-Functional Requirement	● The corresponding is less than 2s ● The personal information of payment is encrypted

2.21 Use-case: Add a new category

Use Case Name	Add a new category
Brief description	This use case describes how the user add extra custom category to the category list
Actor(s)	User
Basic flow	1. User click 'More' option in the list category on the screen 2. System displays custom category screen 3. User entry the label to the field on the screen 4. System validate the label 5. User clicks 'Done' button on the screen
Alternative flow	**Alternative flow 1: Label is duplicated** 1. At step #4, the system displays a message 'label is already added'. 2. The use case continue step #3 in the basic flow

	Alternative flow 2: Undo add a category 1. After step #2, User clicks the 'Back' button to exit the adding category. 2. The use case ends
Pre-conditions	None
Post-conditions	● User successfully add a custom category to the category list and can use that category later on.
Non-Functional Requirement	None

2.22 **Use-case: View transaction**

Use Case Name	View transaction
Brief description	This use case describes how the User can view the history of transactions (category, date and amount) and access transaction details.
Actor(s)	User
Basic flow	1. User navigates to the statistics screen from bottom navigation. 2. User scrolls up to close the statistics screen. 3. System displays the list history of transactions. 4. Users can choose the transaction they want to see detail. 5. The app presents detailed information about the selected transaction, including: a. Transaction type (income or expense). b. Amount. c. Wallet type d. Date and time. e. Category. f. Additional notes (if any). 6. The user can navigate between transactions, viewing details for each other (by slide to left or right)
Alternative flow	**Alternative flow 1: Hide history of transactions** 1. From step #3, user scroll down to minimize the history of transaction 2. The use case ends **Alternative flow 2: Filter Transactions** 1. The user may choose to filter transactions based on criteria such as: a. Date range. b. Transaction type (income, expense). c. Category.
Pre-conditions	● The user is logged in with a registered account. ● User in the statistic screen ● Transaction history data is stored securely and can be retrieved efficiently. ● Transaction history is synced and up-to-date.
Post-conditions	● User successfully observe the history of transactions

	• The user has successfully viewed the details of the selected transaction.
Non-Functional Requirement	• The amount field is blurred for private information • Time for access to the detail transaction not exceed 2s.

2.23 **Use-case: View statistics**

Use Case Name	View statistics
Brief description	This use case describes how the User can track the amount of expense and income in a time period, present a visual statistic that displays an overview using charts
Actor(s)	User
Basic flow	1. User open Dugbet app and clicks statistic (chart) icon in the bottom navigation bar on the screen 2. System navigate the statistic screen 3. The app displays a default summary of total income and total expenses for a specified time period (e.g., current month, last 30 days). 4. User can select time period to change view (default is a week) 5. System displays the chart of income and expense amount corresponding to the time period. 6. The user can interact with the charts, such as tapping on a column, 2D points to see detailed information or expanding detail transaction for category details.
Alternative flow	**Alternative flow 1: Login information is invalid** 1. The use case ends and invoke the use case View transaction **Alternative flow 2: Change Time Period** 2. If the user wants to view the statistic for a different time period: 3. The app provides options to change the time period (e.g., monthly, weekly). 4. The charts are updated dynamically to reflect data for the selected time period. **Alternative flow 3: Customize Chart Appearance** 1. The user may have options to customize the appearance of the charts: 2. Choose between different chart styles (e.g., bar chart instead of columns). 3. Adjust color schemes.
Pre-conditions	• The user is logged in with a registered account. • Transaction data, including income and expense records, is available • Transaction and category data is stored securely and can be retrieved efficiently..
Post-conditions	• User successfully observe the information about expense and income in a time period, present a visual statistic that displays an overview using charts

Non-Functional Requirement	1. The app should ensure that chart data is updated in real-time or with minimal - latency. 2. The charts should be visually appealing and easy to interpret.

2.24 Use-case: Manage financial goal

Use Case Name	Manage financial goal
Brief description	This use case describes how the User can create, view, modify and delete financial goals
Actor(s)	User, Notification system
Basic flow	1. User clicks ‘goal’ icon in the header navigation bar on the screen 2. System navigate the Financial goal screen 3. System display a list of active financial goals 4. User clicks any financial goal tile 5. System navigates to the financial goal detail 6. System display the corresponding information of that financial goal
Alternative flow	**Alternative flow 1: Create financial goal** 1. At step #3 of basic flow, user click on ‘Add’ button on the screen 2. System display adding financial goal screen 3. User entry category, amount, start date, end date, repeat option, wallet to corresponding fields on the screen 4. User click ‘Create’ button at the bottom of the screen 5. The use case continues at step #3 of basic flow **Alternative flow 2: Modify financial goal** 1. At step #6 of basic flow, user click on ‘Edit’ button on the screen 2. System display adding financial goal screen 3. User edit category, amount, start date, end date, repeat option, wallet to corresponding fields on the screen 4. User click ‘Save’ button at the bottom of the screen 5. The use case continues at step #6 of basic flow **Alternative flow 3: Delete financial goal** 1. At step #6 of basic flow, user click on ‘Delete’ button on the screen 2. System show a dialog to verify action 3. User click ‘Delete’ button on the dialog 4. The use case continues at step #3 of basic flow **Alternative flow 4: View history of financial goal** 1. At step #3 of basic flow, user slide to the left 2. System display the history of financial goal
Pre-conditions	• User in the Home screen
Post-conditions	• User able to perform such action: create, view, modify and delete their own financial goals

Non-Functional Requirement	None