EvenUp - Software Design Document (SDGr05)

Group 05
Anshika Gupta - CS22BTECH11007
Yash Patel - CS22BTECH11047
Monish Asawa - MA22BTECH11014
Anirudh Saikrishnan - CS22BTECH11004

April 6, 2025

Contents

1 Unit Testing

Below is the complete list of modules requiring Unit Testing:

Input Subsystem (14 modules)

Module Name	**Type**	**Cohesion Type**
GetUserCredentials	Input	Functional
GetSignUpData	Input	Functional
GetForgotPasswordData	Input	Functional
GetResetPasswordData	Input	Functional
GetGroupCreateData	Input	Functional
GetExpenseData	Input	Functional
GetEditExpenseData	Input	Functional
GetDeleteExpenseRequest	Input	Functional
GetSettleUpRequest	Input	Functional
GetReminderRequest	Input	Functional
GetAnalysisRequest	Input	Functional
GetNightModeToggle	Input	Functional
GetPrivateSplitData	Input	Functional
GetOneTimeSplitConfirm	Input	Functional

Transform Subsystem (16 modules)

Module Name	**Type**	**Cohesion Type**
ForgotPasswordFlow	Transform	Sequential
ResetPassword	Transform	Sequential
CreateGroup	Transform	Functional
JoinGroup	Transform	Functional
ExitGroup	Transform	Functional
DeleteGroup	Transform	Sequential
AddExpense	Transform	Sequential
EditExpense	Transform	Sequential
DeleteExpense	Transform	Sequential
FinalizeSettleUp	Transform	Sequential
SendReminder	Transform	Functional
GenerateSpendingAnalysis	Transform	Functional
FetchTransactionHistory	Transform	Functional
EnableNightMode	Transform	Functional
HandlePrivateSplit	Transform	Sequential
OptimizeSettlements	Transform	Sequential
WebSocketMessageProcessor	Transform	Sequential

Output Subsystem (14 modules)

Module Name	**Type**	**Cohesion Type**
InsertUserRecord	Output	Functional
FetchUserRecord	Output	Functional
InsertGroupRecord	Output	Functional
ModifyGroupMembership	Output	Functional
DeleteGroupRecord	Output	Functional
InsertExpenseRecord	Output	Functional
UpdateExpenseRecord	Output	Functional
RemoveExpenseRecord	Output	Functional
UpdateBalances	Output	Functional
FetchBalances	Output	Functional
StoreSettlementRecord	Output	Functional
FetchTransactions	Output	Functional
SendEmailNotification	Output	Functional
GenerateDisplayView	Output	Functional
WebSocketBroadcaster	Output	Sequential
WebSocketNotifier	Output	Sequential

Total Modules Requiring Unit Testing: 14 (Input) + 16 (Transform) + 14 (Output) = **44 modules**

1.1 Selected Modules for Detailed Unit Testing

Module Type	**Module Name**	**Purpose**	**Risk Factors**
Input	GetUserCredentials	Captures username/password from login screen	First security boundary, input validation
Input	GetExpenseData	Receives expense details for group	Complex data structure, multiple validations
Transform	OptimizeSettlements	Minimizes transaction count in groups	Algorithm complexity, balance calculations
Transform	GenerateSpendingAnalysis	Generates expense analytics	Data aggregation, filtering logic
Transform	DeleteGroup	Removes group and dependencies	High fan-out, data integrity risks
Output	UpdateBalances	Writes balance updates to DB	Critical financial data, high coupling
Output	DeleteGroupRecord	Deletes group from database	Data persistence, cascading effects

1.2 Test Case Specifications

1.2.1 GetUserCredentials (Input Module)

Seq	Condition	Test Data	Expected Result
1	Valid credentials	Username: "yash_patel", Password: "Secure@123"	Returns (username, password) tuple
2	Empty password	Username: "yashP", Password: ""	Throws empty field error
3	SQL injection attempt	Username: "admin'; –", Password: "hack"	Sanitizes input, rejects special chars
4	Maximum length exceeded	Username: $\geq$256-char string, Password: "passwoRd"	Throws error 'Username max length exceeded'

Table 5: GetUserCredentials Test Cases

1.2.2 GetExpenseData (Input Module)

Seq	Condition	Test Data	Expected Result
1	Valid expense	GroupID: 5, Amount: 1000, PaidBy: {UserA: 1000}, Involved: {UserA: 500, UserB: 500}	Returns ExpenseInputData object
2	Negative amount	Amount: -500	Rejects with "Invalid amount" error
3	Sum mismatch	PaidBy: {UserA: 600}, Involved: {UserA: 200, UserB:300}	Detects $200+300 \neq 600$, rejects input
4	Non-member user	Involved: {UserC} (not in group)	Rejects with "Invalid participant" error

Table 6: GetExpenseData Test Cases

1.2.3 OptimizeSettlements (Transform Module)

Seq	Condition	Test Data	Expected Result
1	Simple settlement	Balances: {A: -500, B: +500}	Single transaction: B→A 500, No optimization required
2	Complex network	**Expense1** Payer: A Amount: 1000 Participants: A, B (Split equally) **Expense2** Payer: B Amount: 1000 Participants: B, C (Split equally)	Optimizes to C→A 500
3	Floating-point values	Balances: {A:-333.33, B:+333.33}	Handles decimal precision correctly

Table 7: OptimizeSettlements Test Cases

1.2.4 GenerateSpendingAnalysis (Transform Module)

Seq	Condition	Test Data	Expected Result
1	Monthly analysis	DateRange: 'Last Month'	Correct monthly aggregation
2	Empty dataset	No expenses in date range	Returns "No data" response
3	Multi-category	Expenses in Food(3), Travel(5), Others(2)	Proper category percentages

Table 8: GenerateSpendingAnalysis Test Cases

1.2.5 DeleteGroup (Transform Module)

Seq	Condition	Test Data	Expected Result
1	Valid deletion	GroupID: gr-15 (no debts)	Returns success, cascades deletions
2	Pending debts	GroupID: gr-22 (unsettled 500)	Rejects with "Clear debts first"
3	Non-admin user	Regular user tries to delete	Throws "Admin privileges required"

Table 9: DeleteGroup Test Cases

1.2.6 UpdateBalances (Output Module)

Seq	Condition	Test Data	Expected Result
1	Normal update	Owed: {A: 500}, Owes: {B: 500}	Writes to balances table
2	Precision	Owed: {A: 123.45}	Stores exact decimal(10,2)
3	DB constraint violation	Invalid UserID in owes dict	Rolls back transaction

Table 10: UpdateBalances Test Cases

1.2.7 DeleteGroupRecord (Output Module)

Seq	Condition	Test Data	Expected Result
1	Valid deletion	GroupID: group_42 (exists)	DELETE query succeeds
2	Non-existent group	GroupID: group_999	Returns "Not found" error
3	Orphaned records	GroupID with dangling Foreign Keys	Cascades delete via DB constraints
4	SQL injection	GroupID: "42; DROP TABLE users"	Parameterized query prevents injection

Table 11: DeleteGroupRecord Test Cases

2 Integration Testing

2.1 List of possible module interactions that require Integration Testing

Module 1	→	**Module 2**	**Interaction Type**
GetSignUpData	→	CreateUserAccount	Input → Transform
GetReminderRequest	→	SendReminder	Input → Transform
GetGroupCreateData	→	CreateGroup	Input → Transform
GetForgotPasswordData	→	ForgotPasswordFlow	Input → Transform
GetPrivateSplitData	→	HandlePrivateSplit	Input → Transform
GetExpenseData	→	AddExpense	Input → Transform
GetEditExpenseData	→	EditExpense	Input → Transform
AddExpense	→	OptimizeSettlements	Transform → Transform
EditExpense	→	OptimizeSettlements	Transform → Transform
ForgotPasswordFlow	→	SendEmailNotification	Transform → Transform
CreateUserAccount	→	InsertUserRecord	Transform → Output
ExitGroup	→	ModifyGroupMembership	Transform → Output
SendReminder	→	SendEmailNotification	Transform → Output
CreateGroup	→	InsertGroupRecord	Transform → Output
EditExpense	→	UpdateExpenseRecord	Transform → Output
DeleteExpense	→	RemoveExpenseRecord	Transform → Output
OptimizeSettlements	→	UpdateBalances	Transform → Output
DeleteGroup	→	DeleteGroupRecord	Transform → Output
SettleUp	→	StoreSettlementRecord	Transform → Output
InsertExpenseRecord	→	SendEmailNotification	Output → Output

2.2 Selected Module Interactions for Integration testing

The following table lists 5 Module interactions from the Software Design document, along with the corresponding test case specifications.

Seq No	**Module 1**	**Module 2**	**Interaction Type**
1	GetSignUpData	CreateUserAccount	Input - Transform
2	AddExpense	OptimizeSettlements	Transform - Transform
3	DeleteGroup	DeleteGroupRecord	Transform - Output
4	SettleUp	StoreSettlementRecord	Transform - Output
5	InsertExpenseRecord	SendEmailNotification	Output - Output

Table 13: Module Interaction Table

2.3 Test Case Specifications

Note

Some test cases involve system failures or unexpected conditions that cannot be triggered automatically. These scenarios require manual intervention to simulate failures and observe system behavior.

2.3.1 GetSignUpData → CreateUserAccount

Seq No	Condition to be tested	Test data	Expected result
1	Successful account creation	Name: "John Doe" Email: "john.doe@example.com" Password: "SecurePass123" Username: "john_d"	Account is created successfully, password is hashed, and user record is inserted into the database.
2	User Already Exists	Name: "John Doe" Email: "john.doe@example.com" Password: "SecurePass123" Username: "existing_user"	System detects duplicate username and prompts the user to enter a different one. Account creation is rejected.
3	Password Hashing Fails	Name: "John Doe" Email: "john.doe@example.com" Password: "SecurePass123" Username: "john_d"	Account creation is aborted, system logs an error, and user is notified of a technical issue.
4	Database Insertion Fails	Name: "John Doe" Email: "john.doe@example.com" Password: "SecurePass123" Username: "john_d"	System fails to insert user record into the database, logs an error, and returns a failure response. Account is not created.

Table 14: Test Case Specification Table

2.3.2 AddExpense → OptimizeSettlements

Seq No	Condition to be tested	Test Data	Expected Result
1	Normal Case	Group ID: "group_123" Expense: "Dinner Bill - 1000 Rs" Payer: "User A" Participants: "User A, User B, User C"	Expense is added successfully, balances are updated, transactions are optimized, and WebSocket notification is sent.
2	Database Insertion Fails for Expense Record	Group ID: "group_123" Expense: "Dinner Bill - 1000 Rs" Payer: "User A" Participants: "User A, User B, User C" Simulated database failure	Expense addition is aborted, system logs the failure, and user receives an error message. The OptimizeSettlements function is not called.
3	Optimization Algorithm Fails	Group ID: "group_123" Expense: "Dinner Bill - 1000 Rs" Payer: "User A" Participants: "User A, User B, User C" Simulated failure in 'optimizeSettlements()'	Expense is not added, system logs error, and user is notified with a failure/try again later message.

Table 15: Test Cases for Expense Addition and Optimization

2.3.3 DeleteGroup → DeleteGroupRecord

Seq No	Condition to be tested	Test Data	Expected Result
1	Normal Case – Group deletion when no pending expenses	Group ID: "group_123" Existing Expenses: "None" Call 'delete_group()'	Group is deleted successfully from the database. WebSocket notification is sent to all group members.
2	Group Deletion Attempt When Expenses Still Exist	Group ID: "group_456" Existing Expenses: "exp_3, exp_4" Call 'delete_group()'	Group deletion is aborted and the user receives a message stating that the group cannot be deleted while expenses exist.
3	Database Failure During Group Deletion	Group ID: "group_789" Existing Expenses: "None" Simulated database failure while calling 'delete_group()'	Group deletion fails, system logs the failure, and the user is notified with an error message.

Table 16: Test Cases for Group Deletion Based on Expense Status

2.3.4 SettleUp → StoreSettlementRecord

Seq No	**Condition to be tested**	**Test Data**	**Expected Result**
1	Successful Initialization of Settlement	Payer: “User A” Payee: “User B” Amount: “500 Rs” Action: “Payer initiates settle-up”	Settlement is successfully stored in the database, balances are updated, WebSocket notification is sent, and email confirmation is triggered.
2	Settlement Confirmation by Payee	Payer: “User A” Payee: “User B” Amount: “500 Rs” Action: “Payee confirms settlement”	Settlement is marked as completed in the database, intermediate transactions table is updated, and notifications (WebSocket and email) are sent.
3	Settlement Rejection by Payee	Payer: “User A” Payee: “User B” Amount: “500 Rs” Action: “Payee rejects settlement”	Settlement record is not stored, balances are reverted, and payer receives notification.

Table 17: Test Cases for Settlement and Record Storage

2.3.5 InsertExpenseRecord → SendEmailNotification

Seq No	Condition to be Tested	Test Data	Expected Result
1	Normal Case – Expense successfully inserted and notification sent	Group ID: "group_123" Expense: "Dinner Bill - ₹1000" Payer: "User A" Participants: "User A, User B, User C"	Expense is inserted in database, ExpenseID is returned, and email notifications are sent to all participants.
2	Email Service Fails – Expense added, but email notification fails	Group ID: "group_123" Expense: "Dinner Bill - ₹1000" Payer: "User A" Participants: "User A, User B, User C" Simulated failure in 'SendEmailNotification.send()'	Expense inserted successfully, ExpenseID is returned, but email notifications fail. The system logs an error.
3	Database Insertion Fails	Group ID: "group_123" Expense: "Dinner Bill - ₹1000" Payer: "User A" Participants: "User A, User B, User C"	Expense is not inserted, no ExpenseID is generated, and email notifications are not triggered. An error message is displayed to the user.

Table 18: Test Cases for InsertExpenseRecord and SendEmailNotification

Observations on Module Interactions

An interesting similarity across all module interactions is the clear *trigger-effect* relationship—where the output of one module deterministically prompts an action in the next. This creates a straightforward chain of responsibility, making each interaction testable through clear input-output expectations. However, differences arise in the *type and strength of dependency* between modules. Some interactions involve strict logical conditions before proceeding (like deletions checking pending records), while others represent more optional, side-effect actions (like sending notifications). These variations impact the criticality and scope of testing—ranging from essential correctness validation to user feedback reliability.

3 System Testing

3.1 Selected Use Cases for System Testing

The following table lists the interesting use cases selected from the Software Requirements Specification (SRS) document, along with the corresponding test case specifications and test case generation methods used.

Use Case	**Test Case Specifications**	**Test Case Generation Method**
Sign Up	Valid/Invalid inputs for user registration, including edge cases like empty fields, existing username used to signup, invalid email format, Weak password	Equivalence Partitioning, Boundary Value Analysis
Login	Valid/Invalid credentials, Email Verification complete or incomplete	Equivalence Partitioning, Boundary Value Analysis
Delete Group	Any outstanding balance present or not present, intermediate transactions present or not present, user deleting is admin or not	Equivalence Partitioning, Role-Based Testing
Exit from Group	User exiting from a group with/without outstanding balances, user exiting from the group with/without intermediate transactions	Equivalence Partitioning
Add Expense	Valid/Invalid inputs for adding an expense, including edge cases like empty fields, negative or zero amount, non-numeric amount, very large values, unequal custom splits, very long descriptions, and selecting only self in members.	Equivalence Partitioning, Boundary Value Analysis
Settle Up	Settle up from the group balances page, settle up from the friends page, settle up from the intermediate transactions page	State Transition Testing
View Expenditure Analysis	Valid/Invalid duration filters, no expenses in selected duration	Decision Table Testing
Join Group	Joining an existing group with valid/invalid invite codes, User already a member of the group or not	Equivalence Partitioning

3.2 Test Cases

A collection of sample test cases that will be executed to validate system behavior.

3.2.1 Sign Up

Description	Input	Expected Output
Sign Up with valid details and a strong password	Name: "Yash Patel", Email: "yash@gmail.com", Username: "yash020", Password: "Strong@123"	Successful Registration. An email is sent at the registered email-id for verification, and the user is asked to verify the email.
Sign Up with existing username	Name: "Anshika", Email: "anshika@gmail.com", Username: "anshika020" (Already registered), Password: "Test@123"	Error: username already exists
Sign Up with any empty fields	Name: "", Email: "anshika@gmail.com", Username: "anshika020" Password: ""	Error: Fields cannot be empty
Sign Up with invalid email format	Name: "Yash", Email: "user@", Username: "yash020", Password: "Strong@123"	Error: Enter valid email
Sign Up with weak password	Name: "Yash", Email: "yash@gmail.com", Username: "yash020", Password: "123"	Error: Password does not meet criteria

3.2.2 Login

Description	Input	Expected Output
Login with correct credentials and registered email is verified	Username: "yash_patel", Password: "Secure@123"	Successful Login
Login with incorrect password	Username: "yash_patel", Password: "WrongPass@123"	Error: Invalid credentials
Login with unregistered username	Username: "new_user", Password: "NewUser@123"	Error: username not registered
Login before verifying registered email	Username: "anshika", Password: "Anshika@123" (Email not verified)	Error: Verify your email first
Login with empty fields	Username: "", Password: ""	Error: Fields cannot be empty

3.2.3 Delete Group

Description	Input	Expected Output
Admin of the group deletes group with no outstanding balance and no intermediate transactions	Admin clicks on the group entry and selects Delete option	Group deleted successfully and everyone is sent an analysis of the group expenditures via email.

Delete group with outstanding balance	Admin clicks on the group entry and selects Delete option (Group has outstanding balance)	Error: Cannot delete group with outstanding balances
Delete group with intermediate transactions left	Admin clicks on the group entry and selects Delete option (Group has unsettled transactions)	Error: Intermediate transactions left
Non-admin user trying to delete group	Non-admin user clicks on the group entry and selects Delete option	Error: Only admin can delete the group

3.2.4 Exit from Group

Description	**Input**	**Expected Output**
Exit group such that user does not have any outstanding balances in the group or intermediate transactions in the group	User clicks on the group entry and selects Exit option (No outstanding balances or intermediate transactions)	Successfully exited group
User exits group with outstanding balances in the group	User clicks on the group entry and selects Exit option (Outstanding balances present)	Error: Clear all balances before exiting
User exits group with intermediate transactions present	User clicks on the group entry and selects Exit option (Intermediate transactions present)	Error: Settle all transactions before exiting group

3.2.5 Add Expense

Description	**Input**	**Expected Output**
Add valid expense with equal split	User clicks on 'Add Expense', enters valid description, amount, selects members, selects 'Split equally'	Expense added successfully and amount is equally divided among members
Add valid expense with custom split	User clicks on 'Add Expense', enters valid description, amount, selects members, selects 'Split unequally', and enters valid custom shares	Expense added successfully and amount is divided according to custom shares
Add expense with negative amount	User enters negative amount like -200 while adding expense	Error: Amount must be positive
Add expense with empty fields	User tries to add expense without entering description or amount	Error: Mandatory fields missing
Add expense with invalid amount format	User enters non-numeric value like 'abc' in amount field	Error: Invalid amount

Add expense with custom split such that the sum of the splits is not equal to the amount entered	User enters amount ₹500 but in custom split total of shares entered is ₹400	Error: Sum of splits does not match the total amount
Add expense with zero amount	User enters 0 as amount while adding expense	Error: Amount must be greater than zero
Add expense with self only (no other members)	User enters valid details but selects only themselves in members list	Expense added successfully, full amount owed by self
Add expense with amount having many decimal places	User enters ₹100.99999 as amount	Expense added successfully with amount rounded appropriately
Add expense with very large amount	User enters ₹9999999999 as amount	Error: Very large amount given
Add expense with very large description	User enters very long description	Error: Very large description

3.2.6 Settle Up

Description	**Input**	**Expected Output**
Payer settles up from group balances page	Settle button under the desired balance to be settled in the group balances page is clicked	Transaction is initiated. Balance for the particular group between the two parties is set to zero. An intermediate transaction is added. The payee receives an email notification.
Payee settles up from group balances page	Settle button under the desired balance to be settled in the group balances page is clicked	No change occurs. Transaction is not initiated.
Payer settles up from friends page	Settle button under the desired balance to be settled in the friends page is clicked	Transaction is initiated. Balance between the two parties is set to zero. An intermediate transaction is added. The payee receives an email notification.
Payee settles up from friends page	Settle button under the desired balance to be settled in the friends page is clicked	No change occurs. Transaction is not initiated.
Payee settles up from intermediate transactions page	Settle button under the desired transaction in the intermediate transactions page is clicked	Transaction is completed. Payer is notified via email. Transaction is moved to Logs page.

Payee rejects from intermediate transactions page	Reject button clicked under a transaction in the intermediate transactions page	Transaction is cancelled. Payer is notified via email. Transaction is removed from intermediate transactions page, and the previous balances are restored.

3.2.7 View Expenditure Analysis

Description	**Input**	**Expected Output**
View analysis with valid duration filter	Duration selected: Last 1 Month	Show category-wise expense breakdown for last 1 month
View analysis where no expenses exist in selected duration	Duration selected: Last 1 Month (no expenses present)	Message displayed: No expenses found in selected duration

3.2.8 Join Group

Description	**Input**	**Expected Output**
User enters valid invite code to join a group	User clicks on 'Join Group', enters a correct invite code	Successfully joined the group
User enters invalid invite code	User clicks on 'Join Group', enters an incorrect or random invite code	Error: Invalid invite code
User tries to join a group they are already a part of	User clicks on 'Join Group', enters the invite code of a group they have already joined	Error: Already a member of this group