VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Programming Integration Project – CO3103

FINAL REPORT

Advisor(s): Mr. Le Dinh Thuan
Class: CC01
Group: N8

HO CHI MINH CITY, JANUARY 2026

Member List

No.	Student ID	Full Name
1	2313624	Trần Đỗ Cao Trí
2	2352708	Đinh Cao Thiên Lộc
3	2311883	Nguyễn Thị Kim Loan
4	2352918	Nguyễn Lê Đức Phú
5	2352770	Trần Hà My
6	2153485	Nguyễn Quang Khởi
7	2311987	Nguyễn Song Minh Luân

Contents

1 REQUIREMENT ELICITATION

1.1 Domain Context

In today's group trips or activities, payments are decentralized: different members take turns paying for accommodation, admission tickets, meals or transportation. After the trip, the group often struggles to compile each person's share of the expenses and to settle transfers to the correct recipients. As a result, the need for a shared, transparent, and easy-to-use tool is growing. However, the common practice still relies on manual spreadsheets, scattered notes in chat applications, or even memory. This leads to several limitations: data are not centralized, errors are easy to make, revision history is hard to trace, and real-world situations—such as multiple people jointly paying for a single item—are not well supported. Consolidating all member' expenses typically takes a long time duo to manual reconciliation and writing bank-transfer details, which can cause payments to go to the wrong recipient or be for the wrong amount.

The task is to build a system that standardizes and automates the entire process. Specifically, each trip is managed with a unique ID and a corresponding access link that all members can open to view. Users add the list of participants and record transactions including the description, amount, payer(s), and the list of beneficiaries with allocation weights. The system automatically computes each individual's share. At settlement, it provides a transfer list and allows a designated treasurer (coordinator) to be assigned to collect the funds. At the individual level, the system generates bank-transfer QR codes accordingly: if a person still owes money, their QR directs payment of the exact amount to the treasurer; if a person is to be reimbursed, their QR is used by the treasurer to transfer funds back to them. This approach aligns with the project's core requirements by ensuring transparency, ease of use, and a reduced risk of errors.

1.2 Stakeholders and Needs

Stakeholders	Role	Needs
Event Organizer / Treasurer	Creates and manages events (unique ID/shareable link), manages participants, adds/edits/deletes transactions, assigns a collector, calculates & summarizes expenses, views reports, generates payment QRs, exports data/closes the event	Fast flows, consistent data, easy sharing with the group
Participants	Join via link, update profile, view event info, contribute transactions, view balance summary, view payment QR, request withdrawal, view transaction history, leave event	Transparency, instant access, auditability
Collector	Receives payments per final balances; needs accurate bank info that can be updated so the system can generate standard VietQR codes	Accurate bank info that can be updated so the system can generate standard VietQR codes
Banking/Payment & QR service	Supplies data to generate QRs	VietQR-compliant, high-resolution, fast-generated codes
Database system, File export service	Persists/queries event, participant, and transaction data; encrypts sensitive banking info; exports CSV/PDF	Stable integration, data integrity, good performance

Table 1: Stakeholders, their roles and needs

1.3 Scope of Project

In-Scope:

- Build the event creation & management module: generate a unique Event ID and a shareable link; update/delete events.
- Participant management (target ≥ 50 participants per event): add/edit/remove, store basic profile and optional banking info.
- Transaction management: description, amount, support multiple payers per expense and weighted allocation to beneficiaries.
- Computation & summary: totals paid/benefited and final balances (pay/receive) per participant; view summary reports.
- Collector role: designate a treasurer for the event; store banking information for settlement.
- VietQR payment code generation based on each participant's balance; allow download/share of QR codes.
- Data export (CSV/PDF, UTF-8) and close event to lock further modifications.

- Participant-side features: join via link, update profile, view event details, view summaries & personal QR, contribute expenses, submit withdrawal requests, view history, and leave the event.
- Key performance targets (objective): fast create/update flows, summary computation within a few seconds for small/medium groups, quick QR generation, responsive UI.

Out-of-Scope (Deferred/Post-MVP):

- Automatic payment processing / payment-gateway integration (bank transfers, collection on behalf, automated reconciliation).
- Automatic bank-statement reconciliation; real-time payment confirmation.
- Advanced features beyond core needs: intelligent/ML assistance, multi-currency, dedicated native apps, complex push-notification system.

1.4 Benefits of the System

- **Transparency & auditability:** Centralized records show who paid, who benefited, and final balances; participants can review summaries and history.
- **Fairness & accuracy:** Automatic computation with allocation weights and multiple payers reduces human errors versus manual spreadsheets.
- **Faster settlement via QR:** Per-participant VietQR codes (debt → pay collector; credit → receive from collector), easy to download/share.
- **Time-saving operations.**
- **Scalability & record-keeping:** Supports ≥50 participants per event; export to CSV/PDF UTF-8 for archiving/reconciliation.
- **Sensitive-data protection.**
- **Easy sharing & access.**

2 System Requirements Specification

2.1 Functional Requirements

- **FR1 — Create event**
  The system allows creating a new event, generates a unique Event ID and shareable link bill.thuanle.me/event/{EVENT_ID}, persists it, and shows errors for invalid input.
- **FR2 — Update/Delete event**
  Enable editing event info or deleting an event; confirmation required; success/failure feedback returned.

- **FR3 — Manage participants**
  Add/Edit/Delete participants; optional banking info; prevent deletion if referenced by transactions.
- **FR4 — Assign collector**
  Select a participant as collector and store their banking information for settlement.
- **FR5 — Manage transactions**
  Add/Edit/Delete transactions with: description, amount, multiple payers, and beneficiaries with ratios; validate data before persisting.
- **FR6 — Calculate & summarize expenses**
  Compute total spent, total benefited, and final balance (pay/receive) per participant; allow per-person drill-down and report export.
- **FR7 — View summary report**
  Display summary table (name, spent, benefit, balance), support sort/filter and export CSV/PDF.
- **FR8 — Update collector info**
  Enter/update collector's bank name, account number, account owner; used automatically for payment QR generation.
- **FR9 — Generate VietQR payment codes**
  Generate per-participant QR based on balances: debt → pay collector, credit → receive from collector; allow download/share (ZIP/PDF).
- **FR10 — Export data / Close event**
  Export CSV/PDF (UTF-8, with headers); close event to lock add/edit/delete operations and display Closed status.
- **FR11 — Update personal info**
  User updates profile (phone, address, ...) after login; system verifies they can access their own profile.
- **FR12 — View event information**
  Joined participant views event details (members, transactions, current fund); non-member sees Join Event invitation.
- **FR13 — Join event via link**
  Access bill.thuanle.me/event/{ID_EVENT}, enter basic info (optional bank), system adds user to member list.
- **FR14 — View money distribution summary**
  View spent/benefit/balance; can switch to transaction detail or payment QR.
- **FR15 — View payment QR**
  Show payment details and QR for remaining amount due.

- **FR16 — Contribute transaction**
  Add own expense: amount, multiple payers (if any), benefit ratios; default to equal split when ratios omitted.
- **FR17 — Submit withdrawal request**
  Submit refund/withdrawal with amount, reason, receiving account; system records and notifies organizer for approval.
- **FR18 — View transaction history**
  View chronological transaction list with filter/sort and per-entry details.
- **FR19 — Leave event**
  Remove oneself from the event's participant list.

2.2 Non-Functional Requirements

2.2.1 Performance & Scalability

- Event creation $\leq$ 3s; name supports Unicode.
- Participant updates reflect instantly ($\leq$ 500ms); support $\geq$ 50 participants/event.
- Save/update transaction $\leq$ 2s; UI remains responsive while loading participants.
- Summary calculation $\leq$ 3s for $\leq$ 50 txns & 20 users; result UI $\leq$ 200ms.
- Load summary $\leq$ 2s for $\leq$ 20 users; sorting/filtering $\leq$ 200ms.
- Generate payment QRs $\leq$ 5s for $\leq$ 20 users; each QR $\geq$ 300$\times$300 px.
- Export files $\leq$ 5s for $\leq$ 50 txns & 20 users.
- Event info load $\leq$ 5s; participant summary load/calc $\leq$ 5s; personal QR generated $\leq$ 2s.

2.2.2 Correctness & Data Integrity

- Amounts use two decimal places; ratio computations accurate with real numbers.
- Updates propagate immediately to dependents; deletions preserve integrity (cascade).
- Once closed, disable all modification APIs and show Closed badge.
- Withdrawal requests stored securely and immutable after submission.

2.2.3 Security & Privacy

Encrypt sensitive banking info at rest; validate account number format; authenticate/authorize access to profiles and event pages.

2.2.4 Availability & Error Handling

Clear error messages for DB errors, duplicate IDs, missing data, unavailable QR service; skip participants lacking bank info during QR generation with notices.

2.2.5 Usability & Accessibility

- Real-time input validation; snappy UI (200ms) for sort/filter; mobile-friendly join flow.
- Event name supports Unicode for multilingual input.

2.2.6 Compatibility & Compliance

QRs must follow the VietQR standard encoding account, amount, note; exports use UTF-8 and preserve labels/amounts.

2.2.7 Business Constraints

The system does not execute bank transfers; it computes & generates QRs so users settle the right amounts to the right recipients.

3 Use-case diagram and Use-case scenario

3.1 System Use-case

Figure 1: Use-case for system

3.2 Common usecase

3.2.1 UC-SYSTEM-01: Login

Table 2: Use-case: Login

Field	**Details**
Use-case Name	Login
Use-case ID	UC-SYSTEM-01
Primary Actors	Event Organizer, Participant
Secondary Actors	Database System, Authentication Service
Goal/Purpose	To allow users to securely access the system.
Scope	Travel Expense Splitter Web System
Pre-conditions	User has a registered account in the system.
Post-conditions	**Success:** User is authenticated and redirected to dashboard. **Failure:** Error message is displayed.
Normal Flow	1. User accesses the login page. 2. System displays login form (Email and Password fields). 3. User enters email and password. 4. User clicks "Login" button. 5. System validates credentials against database. 6. System generates session token (JWT) with 7-day expiration. 7. System redirects user based on role: • Event Organizer: Event management dashboard • Participant: List of joined events 8. System displays welcome message.

Continued on next page

Field	Details
Alternative Flow	**A1. Forgot Password:** 1. At Step 2, User clicks "Forgot Password". 2. System displays password reset form. 3. User enters email address. 4. System sends reset link via email. 5. User clicks link and creates new password. **A2. Login with Google:** 1. At Step 2, User clicks "Login with Google". 2. System redirects to Google OAuth. 3. User authorizes and System logs them in.
Exception Flow	**E1. Invalid Credentials:** E1. At Step 5, System detects incorrect email or password. E2. System displays: "Invalid email or password." E3. System limits to 5 attempts per 15 minutes. **E2. Database Error:** E2. At Step 5, System cannot connect to database. E2. System displays: "Unable to process login. Please try again later."
Special Requirements	• Passwords hashed using bcrypt. • Login process completes within 2 seconds. • All endpoints use HTTPS. • Session tokens expire after 7 days of inactivity.

3.2.2 UC-SYSTEM-02: Register

Table 3: Use-case: Register

Field	Details
Use-case Name	Register
Use-case ID	UC-SYSTEM-02
Primary Actors	New User
Secondary Actors	Database System, Email Service
Goal/Purpose	To allow new users to create an account.
Scope	Travel Expense Splitter Web System
Pre-conditions	User does not have an existing account.
Post-conditions	**Success:** New account created and user is logged in. **Failure:** No account created, error message displayed.
Normal Flow	1. User clicks "Register" button. 2. System displays registration form with fields: • Full Name (required) • Email (required) • Password (required, min 8 characters) • Confirm Password (required) • Phone Number (optional) 3. User fills in information. 4. User clicks "Create Account". 5. System validates input data. 6. System checks email does not already exist. 7. System hashes password using bcrypt. 8. System creates new user account with default role "Participant". 9. System sends verification email. 10. System automatically logs in user. 11. System redirects to dashboard.

Continued on next page

Field	Details
Alternative Flow	**A1. Register with Google:** 1. At Step 2, User clicks "Sign up with Google". 2. System redirects to Google OAuth. 3. Google returns user profile (email, name). 4. System auto-fills form with Google data. 5. System creates account (no password needed). 6. Continue from Step 10.
Exception Flow	**E1. Email Already Registered:** E1. At Step 6, System detects email exists. E2. System displays: "This email is already registered. Please login." **E2. Invalid Input:** E2. At Step 5, System detects validation errors. E2. System displays specific error messages: • "Password must be at least 8 characters." • "Invalid email format." • "Passwords do not match." E2. User corrects and resubmits. **E3. Database Error:** E3. At Step 8, System cannot save account. E3. System displays: "Unable to create account. Please try again later."
Special Requirements	• Passwords hashed using bcrypt. • Registration completes within 3 seconds. • Email verification required before creating events. • Default role: "Participant".

3.3 Event Organizer

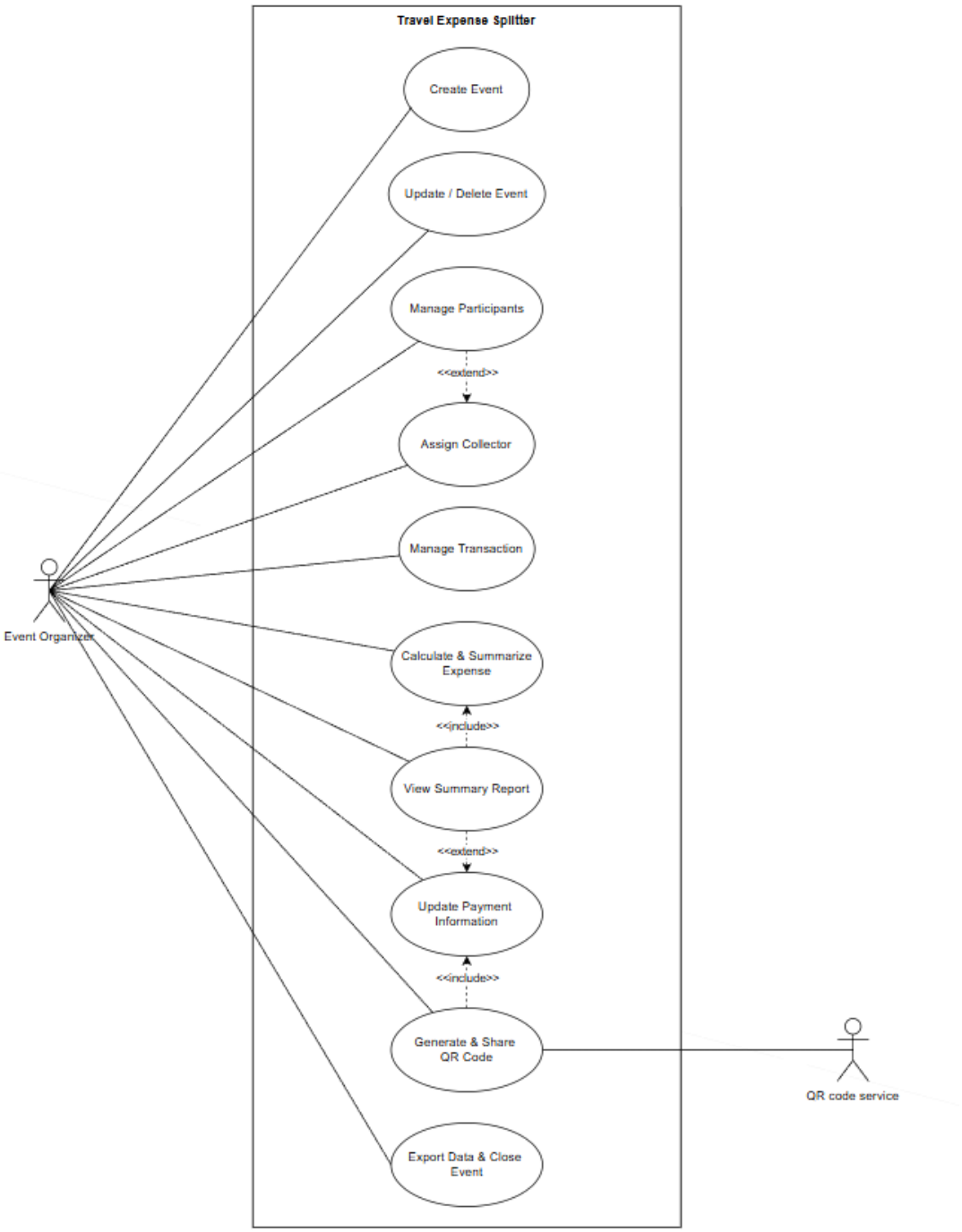

Figure 2: Use-case for event organizer

3.3.1 UC-ADMIN-01: Create Event

Table 4: Use-case: Create Event

Field	Details
Use-case Name	Create Event
Use-case ID	UC-ADMIN-01
Primary Actors	Event Organizer
Secondary Actors	Database System – stores and retrieves event data.
Goal/Purpose	To allow the event organizer to create a new trip or event for managing shared expenses among participants.
Scope	Travel Expense Splitter Web System
Level	User-goal level
Pre-conditions	• The organizer has access to the system interface. • No existing event with the same identifier.
Post-conditions	**Success:** • A new event record is created and stored in the database. • The system generates a unique event ID and shareable link (bill.thuanle.me/event/{EVENT_ID}). **Failure:** • The system displays an error message if event creation fails (e.g., missing name, connection error).

Continued on next page

Field	Details
Normal Flow	1. The Organizer selects the "Create New Event" function. 2. The System displays the event creation form. 3. The Organizer enters event information (name, description, dates, etc.). 4. The Organizer confirms creation. 5. The System validates the data and generates a unique Event ID. 6. The System stores event data in the database. 7. The System displays confirmation and the event access link.
Alternative Flow	• **A1.** Organizer leaves mandatory fields empty: The System prompts the user to complete the missing fields. • **A2.** Organizer cancels before confirmation: No event is created, and the system returns to the previous state.
Exception Flow	• **E1.** Database error: The System displays "Event creation failed. Please try again." • **E2.** Duplicate Event ID generated: The System regenerates a new unique ID and retries saving.
Special Requirements	• The event creation process must complete within 3 seconds. • The event name field must support Unicode characters.

3.3.2 UC-ADMIN-02: Update / Delete Event

Table 5: Use-case: Update / Delete Event

Field	Details
Use-case Name	Update / Delete Event
Use-case ID	UC-ADMIN-02

Continued on next page

Field	Details
Primary Actors	Event Organizer
Secondary Actors	Database System – stores and maintains event data.
Goal/Purpose	To allow the event organizer to modify event information or delete an event entirely.
Scope	Travel Expense Splitter Web System
Level	User-goal level
Pre-conditions	• The event exists in the system. • The organizer has permission to manage it.
Post-conditions	**Success:** • Event information is updated, or an event is removed from the database. **Failure:** • The system shows an error message if update or delete operation fails.

Continued on next page

Field	Details
Normal Flow	**Update:** 1. The Organizer navigates to the event page and selects an existing event. 2. The Organizer selects "Edit Event." 3. The System displays editable event fields. 4. The Organizer modifies the information and confirms. 5. The System validates input and updates the database. 6. The System displays success confirmation. **Delete:** 1. The Organizer navigates to the event page and selects an existing event. 2. The Organizer selects "Delete Event." 3. The System prompts for confirmation. 4. The Organizer confirms deletion. 5. The System removes the event and associated data.
Alternative Flow	• **A1.** Organizer cancels deletion: The operation is aborted, and no data is changed. • **A2.** Invalid input detected during update: The System requests correction before saving.
Exception Flow	• **E1.** Database error during update or delete: The System displays "Unable to process your request." • **E2.** Event already deleted by another user: The System displays "Event not found."
Special Requirements	• Update operations must reflect in all dependent modules immediately. • Deletion must ensure data integrity (cascade delete related participants and transactions).

3.3.3 UC-ADMIN-03: Manage Participants

Table 6: Use-case: Manage Participants

Field	Details
Use-case Name	Manage Participants
Use-case ID	UC-ADMIN-03
Primary Actors	Event Organizer
Secondary Actors	Database System – stores participant data for each event.
Goal/Purpose	To allow the event organizer to add, edit, or remove participants in an event.
Scope	Travel Expense Splitter Web System
Level	User-goal level
Pre-conditions	• The event exists. • The organizer has access rights to modify participant data.
Post-conditions	**Success:** • The participant list is updated accordingly in the database. **Failure:** • Operation fails with a suitable error message (e.g., participant used in transaction).

Continued on next page

Field	Details
Normal Flow	**Add:** 1. The Organizer selects "Add Participant." 2. The System displays participant input form. 3. The Organizer enters participant name and optional bank info. 4. The Organizer confirms the addition. 5. The System stores the new participant record in the database. **Edit:** 1. The Organizer selects an existing participant. 2. The Organizer edits details and confirms changes. 3. The System updates participant information. **Delete:** 1. The Organizer selects "Delete Participant." 2. The System prompts for confirmation. 3. The Organizer confirms deletion. 4. The System removes the participant from the list.
Alternative Flow	• **A1.** Attempt to delete participant linked to transactions: The System shows "Cannot delete participant with existing transactions."
Exception Flow	• **E1.** Database connection failure: The System displays "Unable to save participant information." • **E2.** Duplicate participant name detected: The System displays "Participant already exists."
Special Requirements	• Participant list updates must reflect instantly without page reload (under 500ms). • The system must support at least 50 participants per event.

3.3.4 UC-ADMIN-04: Assign Collector

Table 7: Use-case: Assign Collector

Field	Details
Use-case Name	Assign Collector
Use-case ID	UC-ADMIN-04
Primary Actors	Event Organizer
Secondary Actors	Database System – updates collector information linked to an event.
Goal/Purpose	To allow the event organizer to assign one participant as the money collector who will receive payments from others.
Scope	Travel Expense Splitter Web System
Level	User-goal level
Pre-conditions	• Events and participants exist in the system. • At least one participant has valid banking information.
Post-conditions	**Success:** • Collector information is stored in the database and associated with the event. **Failure:** • The system displays an error if the collector cannot be assigned.
Normal Flow	1. The Organizer navigates to the "Collector" section in the event page. 2. The System displays a list of participants. 3. The Organizer selects one participant as the collector. 4. The Organizer verifies or enters bank account details. 5. The Organizer confirms assignment. 6. The System saves collector information and displays confirmation.

Continued on next page

Field	Details
Alternative Flow	• **A1.** Organizer changes collector → The System updates the new collector and overwrites the previous one. • **A2.** Organizer adds missing bank info before confirmation.
Exception Flow	• **E1.** Selected participant does not exist → The System displays "Invalid collector." • **E2.** Database error during save → The System displays "Failed to assign collector. Please retry."
Special Requirements	• Collector change must update corresponding QR code generation logic immediately. • The system must validate the account number format before saving.

3.3.5 UC-ADMIN-05: Manage Transactions

Table 8: Use-case: Manage Transactions

Field	Details
Use-case Name	Manage Transactions
Use-case ID	UC-ADMIN-05
Primary Actors	Event Organizer
Secondary Actors	Database System – stores and updates transaction data.
Goal/Purpose	To allow the event organizer to add, modify, or delete transactions related to an event.
Scope	Travel Expense Splitter Web System
Level	User-goal level
Pre-conditions	• The event exists in the system. • The participant list has been defined.

Continued on next page

Field	Details
Post-conditions	**Success:** • Transaction information is saved, modified, or removed successfully. **Failure:** • System notifies the user of the issue (e.g., invalid input or database failure).
Normal Flow	1. **Add:** (a) The Organizer selects "Add Transaction." (b) The System displays the transaction input form. (c) The Organizer enters transaction details (description, amount, payer(s), beneficiaries with ratios). (d) The Organizer confirms addition. (e) The System validates and saves the transaction to the database. 2. **Edit:** (a) The Organizer selects an existing transaction. (b) The Organizer updates details and confirms. (c) The System updates the database and displays confirmation. 3. **Delete:** (a) The Organizer selects a transaction to delete. (b) The System prompts for confirmation. (c) The Organizer confirms deletion. (d) The System removes the transaction from the database.
Alternative Flow	• **A1.** Organizer enters invalid amount (e.g., negative or zero) → The System prompts correction. • **A2.** Organizer cancels action before saving → No change is made.

Continued on next page

Field	Details
Exception Flow	• **E1.** Database connection lost → The System displays "Unable to save transaction. Please try again." • **E2.** Data inconsistency detected (payer not in event) → The System displays "Invalid participant selection."
Special Requirements	• The system must process transaction save/update within 2 seconds. • Amount values must be validated to two decimal places. • The interface should remain responsive while loading participant lists.

3.3.6 UC-ADMIN-06: Calculate and Summarize Expense

Table 9: Use-case: Calculate and Summarize Expense

Field	Details
Use-case Name	Calculate and Summarize Expense
Use-case ID	UC-ADMIN-06
Primary Actors	Event Organizer
Secondary Actors	Database System – stores and queries information of participants and transactions.
Goal/Purpose	To help the organizer generate a spending summary for the entire event, including: • The total amount each participant has paid. • The total amount each participant has benefited from. • The amount each participant still owes or should receive. The final result is a fair and transparent expense summary for the whole group.
Scope	Travel Expense Splitter Web System
Level	User-goal level (user's business objective)

Continued on next page

Field	Details
Pre-conditions	• The event has been successfully created. • The list of participants and transactions has been fully entered. • All stored data is valid (no missing payer or beneficiary, all amounts > 0).
Post-conditions	**Success:** • The system displays a summary table including: – Total amount spent by each participant. – Total amount each participant benefited from. – Remaining amount each participant must pay or receive. • The summary data is temporarily stored for display or report export. **Failure:** • The system displays an error message (e.g., missing data, invalid transaction, or no participants found).

Continued on next page

Field	Details
Normal Flow	1. The Organizer selects the event to summarize → The System loads the event, participants, and transaction data. 2. The Organizer chooses the "Calculate Summary" function → The System begins processing calculations. 3. The System calculates the total amount spent by each participant (based on payers). 4. The System calculates the amount each participant benefited from (based on benefit ratios). 5. The System compares the difference between the amount spent and the amount benefited. 6. The Organizer views the calculation results → The System displays a summary list including: name, amount spent, amount benefited, and amount to pay/receive. 7. The Organizer can save or export the summary results → The System saves the summary state or allows exporting as a CSV/PDF file.
Alternative Flow	• **A1.** Organizer wants to view detailed transactions: In Step 6, the Organizer clicks on a participant's name → The System displays a detailed list of that participant's related transactions. • **A2.** Organizer wants to adjust benefit ratios: Before Step 2, the Organizer can return to edit transaction data, then re-run the "Calculate Summary" process.
Exception Flow	• **E1.** No transaction data available: When "Calculate" is clicked, the System displays an error message: "No transactions found for this event." • **E2.** Transaction contains negative amount or missing beneficiaries: The System stops the calculation and shows a specific error message for correction. • **E3.** Database connection error: The System displays: "Unable to access database. Please try again later."

Continued on next page

Field	Details
Special Requirements	• Calculation results must be processed within < 3 seconds for up to 50 transactions and 20 participants. • The result display interface must respond instantly (within 200 ms after computation). • The system must ensure absolute accuracy in ratio calculations (using real numbers with two decimal places).

3.3.7 UC-ADMIN-07: View Summary Report

Table 10: Use-case: View Summary Report

Field	Details
Use-case Name	View Summary Report
Use-case ID	UC-ADMIN-07
Primary Actors	Event Organizer
Secondary Actors	Database System – stores and provides access to event summary data.
Goal/Purpose	To allow the event organizer to view the overall summary report after calculation, including each participant's total spending, benefits, and balance (amount owed or to be received).
Scope	Travel Expense Splitter Web System
Level	User-goal level
Pre-conditions	• The event must already exist. • The expense calculation (UC-ADMIN-06) has been completed successfully. • Summary data has been generated and stored temporarily or permanently in the database.

Continued on next page

Field	Details
Post-conditions	**Success:** • The system displays the event summary report, including: – Participant name – Total paid amount – Total benefit – Balance (positive = receive money, negative = pay more) • The report can be viewed, exported, or printed. **Failure:** • The system shows an appropriate error message if no summary data is found or cannot be loaded.
Normal Flow	1. The Organizer navigates to the event page → The System retrieves event and participant information. 2. The Organizer selects "View Summary Report" function → The System loads the latest calculation results. 3. The System displays the summary table, including each participant's spending, benefit, and balance. 4. The Organizer can sort or filter the data (e.g., by amount, name) → The System refreshes the displayed data accordingly. 5. The Organizer may choose to export the report → The System offers export options (CSV, PDF).
Alternative Flow	• **A1.** Organizer wants to view individual participant details: At Step 3, clicking a participant's name opens a detailed breakdown of that person's transactions. • **A2.** Organizer wants to recalculate before viewing: Before Step 2, the Organizer can trigger "Recalculate Summary" (UC-ADMIN-06) to ensure up-to-date data.

Continued on next page

Field	Details
Exception Flow	• **E1.** No summary data available: The System displays: "No summary data found. Please perform calculation first." • **E2.** Database connection failure: The System shows: "Unable to load summary. Please try again later." • **E3.** Data inconsistency detected: The System notifies: "Invalid or incomplete summary data detected."
Special Requirements	• The summary table must load in < 2 seconds for ≤ 20 participants. • The interface must remain responsive while sorting or filtering (< 200ms delay). • Exported reports must maintain consistent numeric formatting (2 decimal places).

3.3.8 UC-ADMIN-08: Update Collector Information

Table 11: Use-case: Update Collector Information

Field	Details
Use-case Name	Update Collector Information
Use-case ID	UC-ADMIN-08
Primary Actors	Event Organizer
Secondary Actors	Database System – stores collector information (bank name, account number, account owner).
Goal/Purpose	Allow the event organizer to specify or update the payment collector's information, which will be used later to generate payment QR codes for participants to transfer money accurately.
Scope	Travel Expense Splitter Web Application
Level	User-goal level

Continued on next page

Field	Details
Pre-conditions	• The event has been created successfully. • Expense calculation has been completed or event data is ready for settlement. • The organizer has permission to edit collector information.
Post-conditions	**Success:** • The collector's banking information is saved in the database. • The information will be used automatically when generating payment QR codes. **Failure:** • System displays an error if required fields (e.g., account number, bank name) are missing or invalid.
Normal Flow	1. The Organizer opens the "Collector Information" section of the event → The System displays current collector data (if any). 2. The Organizer enters or edits collector info: bank name, account number, and account owner → The System validates input format (non-empty, valid account number). 3. The Organizer clicks "Save" → The System saves data to the database. 4. The System confirms the update → Displays message: "Collector information has been successfully updated."
Alternative Flow	• **A1.** Organizer wants to remove collector info: At Step 2, Organizer clicks "Remove Collector Info." The System clears stored collector data after confirmation. • **A2.** Organizer selects an existing participant as collector: The System automatically fills in that participant's bank information (if available).

Continued on next page

Field	Details
Exception Flow	• **E1.** Missing or invalid input: The System highlights missing fields and shows: "Please enter all required collector information." • **E2.** Database connection error: The System shows: "Unable to update collector info. Please try again later."
Special Requirements	• The System must validate all input fields in real-time (no page reload). • Sensitive banking info must be encrypted before storage. • Update operation should complete in under 2 seconds.

3.3.9 UC-ADMIN-09: Generate Payment QR Code

Table 12: Use-case: Generate Payment QR Code

Field	Details
Use-case Name	Generate Payment QR Code
Use-case ID	UC-ADMIN-09
Primary Actors	Event Organizer
Secondary Actors	• Banking / Payment System – provides information for generating QR codes. • Database System – stores participant and payment data.
Goal/Purpose	To generate payment QR codes that allow participants to easily transfer or receive money based on the final summary. • If a participant owes money, the QR code directs them to transfer the exact amount to the organizer. • If a participant should receive money, the QR code contains their banking info so others can transfer money to them.
Scope	Travel Expense Splitter Web System
Level	User-goal level

Continued on next page

Field	Details
Pre-conditions	• Expense summary (UC-ADMIN-06) must be successfully calculated. • Organizer and participant banking details are valid and available. • The system has access to a payment/QR generation service.
Post-conditions	**Success:** • The system generates and displays QR codes for: – The organizer (for collecting payments). – Each participant (for paying or receiving money). • QR codes can be downloaded or shared directly. **Failure:** • The system notifies the organizer if QR generation fails due to invalid or missing data.
Normal Flow	1. The Organizer opens the summary report page → The System retrieves event summary data. 2. The Organizer clicks "Generate QR Codes." → The System validates participant and payment information. 3. The System determines each participant's balance (positive = receive, negative = pay). 4. For each participant, the System generates a QR code: • If balance < 0 → QR directs payment to Organizer's bank info. • If balance > 0 → QR contains Participant's bank info. 5. The System displays all generated QR codes in a table → Each row includes name, balance, and QR image. 6. The Organizer can download QR codes or share the report → The System provides download and share options (ZIP/PDF).

Continued on next page

Field	Details
Alternative Flow	• **A1.** Organizer generates QR for one specific participant only: At Step 2, the Organizer selects a participant → The System generates and displays only that QR code. • **A2.** Organizer updates bank info before generating: Before Step 2, the Organizer edits the participant's bank details → Then regenerates QR.
Exception Flow	• **E1.** Missing or invalid bank info: The System skips that participant and shows: "Missing banking details for [Name]." • **E2.** Payment service unavailable: The System shows: "Cannot generate QR codes. Please try again later." • **E3.** Database error: The System displays: "Error retrieving payment data."
Special Requirements	• Each QR code must encode correct payment info (account, amount, note) following VietQR standard. • QR generation must complete in < 5 seconds for ≤ 20 participants. • All generated QR codes must maintain high resolution for scanning ($\geq$ 300x300 px). • Exported file (ZIP or PDF) must preserve each QR's label and amount.

3.3.10 UC-ADMIN-10: Export Data / Close Event

Table 13: Use-case: Export Data / Close Event

Field	Details
Use-case Name	Export Data / Close Event
Use-case ID	UC-ADMIN-10
Primary Actors	Event Organizer

Continued on next page

Field	Details
Secondary Actors	• Database System – provides stored event data for export. • File Export Service – generates downloadable CSV or PDF files.
Goal/Purpose	Allow the organizer to finalize the event by either: • Exporting all event data (participants, transactions, final balances) for record keeping, or • Closing the event to prevent further modification.
Scope	Travel Expense Splitter Web Application
Level	User-goal level
Pre-conditions	• The event exists and contains valid participant and transaction data. • Expense calculation and summary have been completed. • The organizer is authenticated and authorized to finalize the event.
Post-conditions	**Success:** • A data file (CSV or PDF) is generated and downloaded to the organizer's device. • The event status changes to **Closed**, preventing further edits. **Failure:** • The system notifies the organizer if the export fails or data is missing.

Continued on next page

Field	Details
Normal Flow	1. The Organizer opens the event summary page → The System displays event overview and available actions. 2. The Organizer clicks "Export Data / Close Event." → The System shows dialog with two options: "Export Data" or "Close Event." label*=1. Organizer selects **Export Data** and chooses file format (CSV or PDF) → The System generates the file and starts download. lbbel*=2. Organizer selects **Close Event** → The System requests confirmation before locking further edits. 3. Organizer confirms the action → The System performs export or status update. 4. The System displays a confirmation message → "Event data exported successfully." or "The event has been closed."
Alternative Flow	• **A1.** Organizer wants to export only specific data: At Step 2, selects custom export options (e.g., only transactions, only participant info). The System generates filtered file accordingly. • **A2.** Organizer cancels the close action: At Step 3b, clicks "Cancel." The System aborts closing and keeps the event open.
Exception Flow	• **E1.** Missing calculated data: The System shows: "Please complete the expense calculation before exporting." • **E2.** Export service failure: The System notifies: "Unable to generate file. Please try again." • **E3.** Database access error: The System shows: "Failed to retrieve event data."

Continued on next page

Field	Details
Special Requirements	• File export must complete within 5 seconds for up to 50 transactions and 20 participants. • Exported files should use UTF-8 encoding and include headers for readability. • Once an event is closed, all modification-related APIs (add/edit/delete) are disabled. • The system must display a clear **Closed** status badge on the event page.

3.4 Participant

This use case focuses on the function that the participants can perform. Provides a transparent, instant, and auditable system for managing communal finances. Specifically, the following use cases define the participant's journey from entering a shared expense to viewing their final, balanced settlement summary.

Figure 3: Use-case for participant

3.4.1 UC-PARTICIPANT-01: Update Personal Information

Table 14: Use-case: Update Personal Information

Field	Details
Use-case Name	Update Personal Information
Use-case ID	UC-PARTICIPANT-01
Primary Actors	Participant
Secondary Actors	Database System
Goal/Purpose	To allow the participant to successfully modify and save their personal profile information within the application.
Scope	Travel Expense Splitter Web System
Pre-conditions	The User is logged in and has navigated to the Profile area.
Post-conditions	The User's profile information is updated and committed to the database.
Normal Flow	1. Participant clicks on the "Update personal information" on the profile. 2. The System verifies the participant's authentication and authorization (ensures the user is logged in and permitted to view their own profile). 3. The System retrieves the participant's current personal data from the database. 4. The System displays the "Edit Profile" form, pre-filled with the current information. 5. The Participant reviews the current data and makes changes to one or more fields that includes phone number, address and social security number. 6. The Participant clicks the "Save Changes". 7. The System performs input validation on the submitted data. 8. The System updates the participant's record in the database with the new, validated information. 9. The System displays a confirmation message.

Continued on next page

Field	Details
Alternative Flow	At Step 4, the User decides not to save changes and clicks a "Cancel" or "Back" button. The System returns the user to the previous page without saving any modifications.
Exception Flow	6a. The system prints a message if the information input is not validated.
Special Requirements	None

3.4.2 UC-PARTICIPANT-02: View Event Information

Table 15: Use-case: View Event Information

Field	Details
Use-case Name	View Event Information
Use-case ID	UC-PARTICIPANT-02
Primary Actors	Participant
Secondary Actors	Share-Bill System
Goal/Purpose	The participant wants to view the details of an event they have joined, including the member list, transactions, and the current total fund.
Scope	Travel Expense Splitter System – web application bill.thuanle.me
Pre-conditions	• The participant has already joined a valid event. • The system has data on the member list and transactions for the event.
Post-conditions	• Success: Event details are displayed accurately to the participant. • Failure: The system displays an error message if the event does not exist or data cannot be retrieved.

Continued on next page

Field	Details
Normal Flow	1. The participant accesses the event page they have joined (bill.thuanle.me/event/{ID_EVENT}). 2. The system verifies the user is on the event's member list. 3. The system retrieves data: event name, organizer, member list, transactions, current total fund. 4. The system displays all event information on the user interface. 5. The participant can choose to view transaction details or switch to other tabs (e.g., summary, payment QR).
Alternative Flow	A1. User viewing the event has not joined: → The system displays the "Join Event" invitation page instead of the details. A2. Large event dataset: → The system loads data using pagination or lazy loading to optimize performance.
Exception Flow	E1. Event does not exist or has been deleted: → The system displays an error message "Event does not exist or has been closed." E2. Server error / network connection issue: → Displays message "Could not load event information, please try again later."
Special Requirements	• Data loading time should not exceed 5 seconds.

3.4.3 UC-PARTICIPANT-03: Join Event

Table 16: Use-case: Join Event

Field	Details
Use-case Name	Join Event
Use-case ID	UC-PARTICIPANT-P03
Primary Actors	Participant
Secondary Actors	Share-Bill System

Continued on next page

Field	Details
Goal/Purpose	The participant wants to join a bill-splitting event to view information, contribute transactions, and track the fund summary.
Scope	Travel Expense Splitter Web System – web application bill.thuanle.me
Pre-conditions	• The event has been successfully created by the organizer. • The organizer shares the participation link or code with the user.
Post-conditions	• Success: The participant is added to the event's member list and can access information and perform actions within the event. • Failure: The system displays an error message (e.g., invalid link, event closed).
Normal Flow	1. The Participant accesses the event link (bill.thuanle.me/event/{ID_EVENT}). 2. The System displays the event invitation interface. 3. The Participant enters basic personal information (name, bank info - optional). 4. The Participant clicks "Join Event". 5. The System validates the information and adds the participant to the event's member list. 6. The System displays the event information page.
Alternative Flow	A1. User has previously joined the event: → The system detects the user already exists in the list and automatically redirects to the event page (skipping the information entry step).
Exception Flow	E1. Invalid link or event deleted: → The system displays an error message "Event does not exist or has been closed." E2. Network connection / server error: → Displays message "Cannot connect to the server, please try again later."

Continued on next page

Field	Details
Special Requirements	• User-friendly interface, accessible from mobile devices.

3.4.4 UC-PARTICIPANT-04: View Money Distribution Summary

Table 17: Use-case: View Money Distribution Summary

Field	Details
Use-case Name	View Money Distribution Summary
Use-case ID	UC-PARTICIPANT-P04
Primary Actors	Participant
Secondary Actors	Share-Bill System
Goal/Purpose	The participant wants to see the final summary of the amount they paid, their share, and the amount they still owe or are due to receive.
Scope	Travel Expense Splitter System – web application bill.thuanle.me
Pre-conditions	• The participant has joined a valid event. • The event has transaction data (at least one expense / contribution has been made). • The system has performed an automatic calculation, or the organizer has initiated the summary.
Post-conditions	• Success: The participant can view their personal and the group's summary (who owes whom). • Failure: The system displays an error message if there is no data or the calculation fails.

Continued on next page

Field	Details
Normal Flow	1. The participant accesses the event page. 2. Selects the "Balance Summary" tab or function. 3. The system retrieves all transaction data for the event. 4. The system calculates (if not already available): total amount spent by each person, their share, and the final difference. 5. The system displays the summary table: member name, total paid, total share, amount to pay or receive. 6. The participant views the results and can switch to see transaction details or the payment QR code.
Alternative Flow	A1. Event summary is not yet finalized: → The system displays a message "Temporary summary – data is subject to change." A2. Summary data has been updated by the organizer: → The system displays the final summary with the finalization date.
Exception Flow	E1. No transactions: → The system displays a message "No transactions available to summarize." E2. Calculation error or invalid data: → Displays message "Could not display summary, please try again later."
Special Requirements	• Data loading and calculation time should not exceed 5 seconds.

3.4.5 UC-PARTICIPANT-05: View Payment QR Code

Table 18: Use-case: View Payment QR Code

Field	Details
Use-case Name	View Payment QR Code
Use-case ID	UC-PARTICIPANT-P05

Continued on next page

Field	Details
Primary Actors	Participant
Secondary Actors	Database System
Description	To generate and display a dynamic QR code that allows the participant to settle their outstanding balance.
Scope	Travel Expense Splitter Web System
Pre-conditions	• The User is a registered participant in the selected event. • The User has updated all information.
Post-conditions	The System successfully displays the payment details and an active QR code representing the remaining amount due.
Normal Flow	1. User selects the desired event from their event list. 2. Participant clicks on the "View payment information" on the event setting. 3. The System checks the transaction history for the specific order. 4. The System calculates the Remaining Amount Due. 5. The System displays the Payment Information including Total Amount Required, Amount Already Paid, Remaining Amount Due, A New QR Code is generated for the Remaining Amount Due.
Alternative Flow	None
Exception Flow	3a. Using invalid sign-in credentials results in an error message.
Special Requirements	The QR code must be generated within 2 seconds.

3.4.6 UC-PARTICIPANT-06: Contribute Transaction

Table 19: Use-case: Contribute Transaction

Field	Details
Use-case Name	Contribute Transaction

Continued on next page

Field	Details
Use-case ID	UC-PARTICIPANT-P06
Primary Actors	Participant
Secondary Actors	Share-Bill System
Goal/Purpose	The participant wants to record an expense or their contribution to the event's shared fund.
Scope	Travel Expense Splitter System – web application bill.thuanle.me
Pre-conditions	• The participant has joined a valid event. • The event is still open, allowing new transactions. • The user has permission to contribute (not locked).
Post-conditions	• Success: The transaction is saved to the system; the total expenses and share distribution among members are updated. • Failure: The system does not save the transaction and displays an error message (e.g., incorrect format, missing information).
Normal Flow	1. The participant opens the event page and selects "Add Transaction". 2. The system displays the form to enter new transaction details. 3. The participant enters: description, amount, selects the payer(s) (can be multiple), and the share for each member (e.g., 1, 1, 1.5, 3). 4. The participant clicks "Confirm Contribution". 5. The system validates the data ($amount > 0$, members exist in the event). 6. The system saves the transaction information and updates the total fund. 7. The system displays a "Transaction added successfully" message and updates the transaction list displayed for all members.

Continued on next page

Field	Details
Alternative Flow	A1. Contribution with multiple payers: → The participant selects multiple payers; the system automatically divides the contribution proportionally. A2. Share distribution not entered: → The system defaults to splitting the cost equally among all members.
Exception Flow	E1. Incorrect amount format or missing required information: → The system displays an error message and requests re-entry. E2. Network connection loss or server error: → The system displays "Cannot add transaction, please try again later."
Special Requirements	• Ensure data integrity.

3.4.7 UC-PARTICIPANT-07: View Transaction History

Table 20: Use-case: View Transaction History

Field	Details
Use-case Name	View Transaction History
Use-case ID	UC-PARTICIPANT-P07
Primary Actors	Participant
Secondary Actors	Database System
Description	To allow the participant to securely access and view chronological list of their financial transaction within the system.
Scope	Travel Expense Splitter Web System
Pre-conditions	• The Participant must be logged in. • The participant must have at least one completed transaction recorded in the database.
Post-conditions	The system displays a filtered, sortable list of the participant's transactions.

Continued on next page

Field	Details
Normal Flow	1. The Participant navigates to the "History Transactions" section of the system. 2. The system verifies the Participant's authentication and authorization. 3. The System retrieves the complete chronological list of the User's relevant financial transactions from the database 4. The System displays the Transaction History interface, presenting the transactions in a list format. 5. The System ensures the following key details are visible for each entry: Date/Time, Description, Amount, and Type (Debit/Credit). 6. The User reviews the transaction list and can optionally use features like sorting (by date/amount) or filtering.
Alternative Flow	• At Step 3, the System finds no transactions for the User. The System displays the history interface (Step 4) but shows an "empty state" message. • At Step 6, the User applies a filter (e.g., only show transactions from "Event A"). The System performs a new database query (Step 3) and re-renders the history list with the filtered results (Step 4) • At Step 5, the User clicks on a specific transaction entry. The System displays a detailed view showing all relevant information (e.g., full description, associated event, linked participants).
Exception Flow	7a. At Step 2, the User's session has expired. The System redirects the User to the Login screen. 7b. At Step 3, the database connection fails or the query times out. The System displays an error message.
Special Requirements	None

3.4.8 UC-PARTICIPANT-08: Leave Event

Table 21: Use-case: Leave Event

Field	Details
Use-case Name	Leave Event
Use-case ID	UC-PARTICIPANT-P08
Primary Actors	Participant
Secondary Actors	Database System
Description	To allow the participant to successfully remove themselves from the event's participant list.
Scope	Travel Expense Splitter Web System
Pre-conditions	The User is logged in and currently a member of the event.
Post-conditions	The User's participation record is removed from the event and they can no longer access the event details.
Normal Flow	1. The Participant navigates to their list of current events. 2. The Participant chooses the specific event they want to leave. 3. The System displays the Event Information page. 4. The Participant locates and clicks the "Leave Event" button. 5. The System displays a confirmation message. 6. The Participant confirms the action. 7. The System executes the database transaction to remove the Participant from the event's participant list. 8. The System displays a successful message.
Alternative Flow	At Step 6, the User clicks "No, Cancel" or closes the confirmation dialog. The system closes the message and returns the User to the Event Information page.
Exception Flow	7a. The Database System fails to update the record. The system displays a generic error message.
Special Requirements	None

4 System Modeling

4.1 State Diagram

4.1.1 User State Diagram

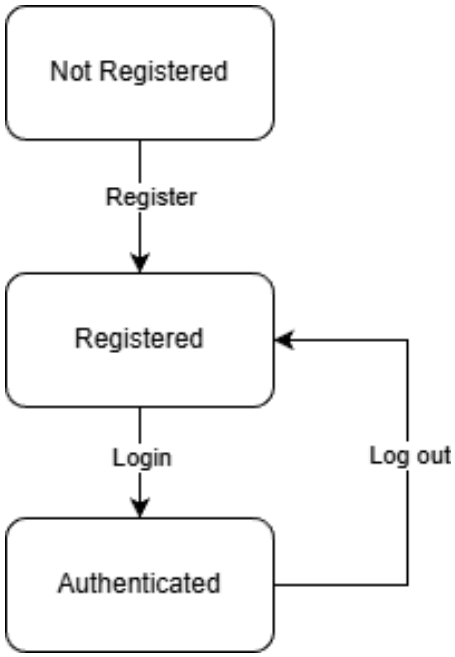

Figure 4: User State Diagram

This state diagram represents the user authentication lifecycle in the system.
A user starts in the Not Registered state and transitions to Registered after successfully creating an account.
From the Registered state, the user can log in to become Authenticated and access system features.
When the user logs out, the system ends the session and returns the user to the Registered state.

4.1.2 Event State Diagram

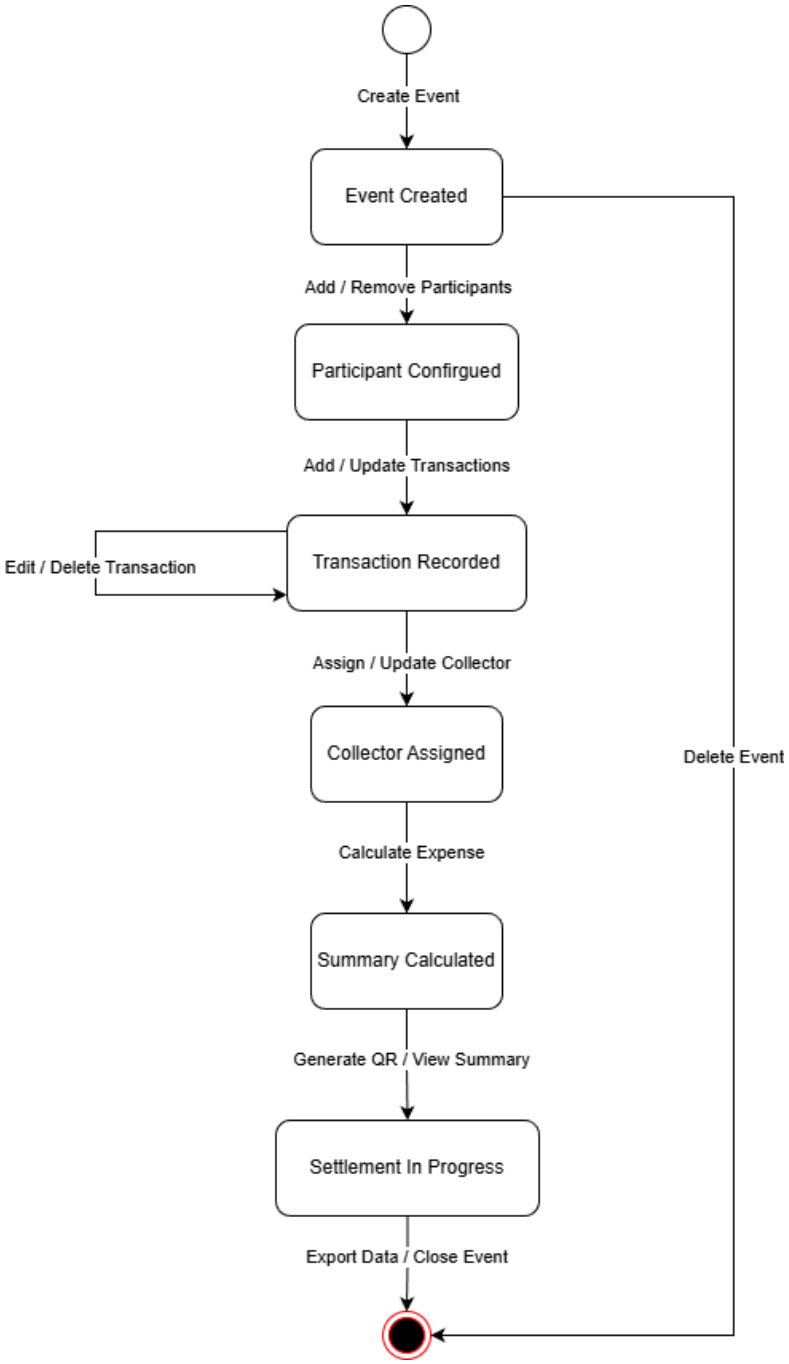

Figure 5: Event State Diagram

This state diagram models the life cycle of an event from creation to closure.
After an organizer creates an event, participants are added or removed until the participant list is configured.
Expenses are then recorded (with an edit/delete loop allowed while transactions exist), and a collector is assigned before calculating the summary.
Once the summary is calculated the system moves into settlement in progress where QR/payment actions occur, and finally the event can be exported/closed.
A delete transition is allowed only in the early stages (before transactions are recorded), while later stages become read-only to preserve financial records.

4.1.3 Participants State Diagram

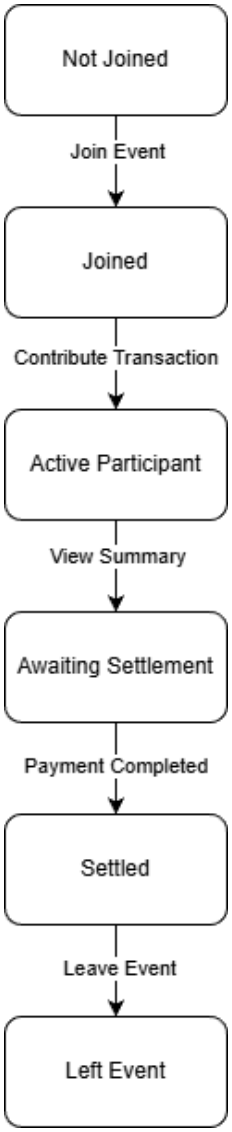

Figure 6: Participants State Diagram

This state diagram describes the life cycle of a participant within an event. Initially, the user is in the Not Joined state and transitions to Joined after successfully joining the event.
By contributing transactions, the participant becomes an Active Participant and can view expense summaries, moving to Awaiting Settlement.
Once payment is completed, the participant reaches the Settled state, indicating all financial obligations are resolved.
Finally, the participant may choose to leave the event, transitioning to the Left Event state, which ends their participation.

4.2 System Sequence Diagram

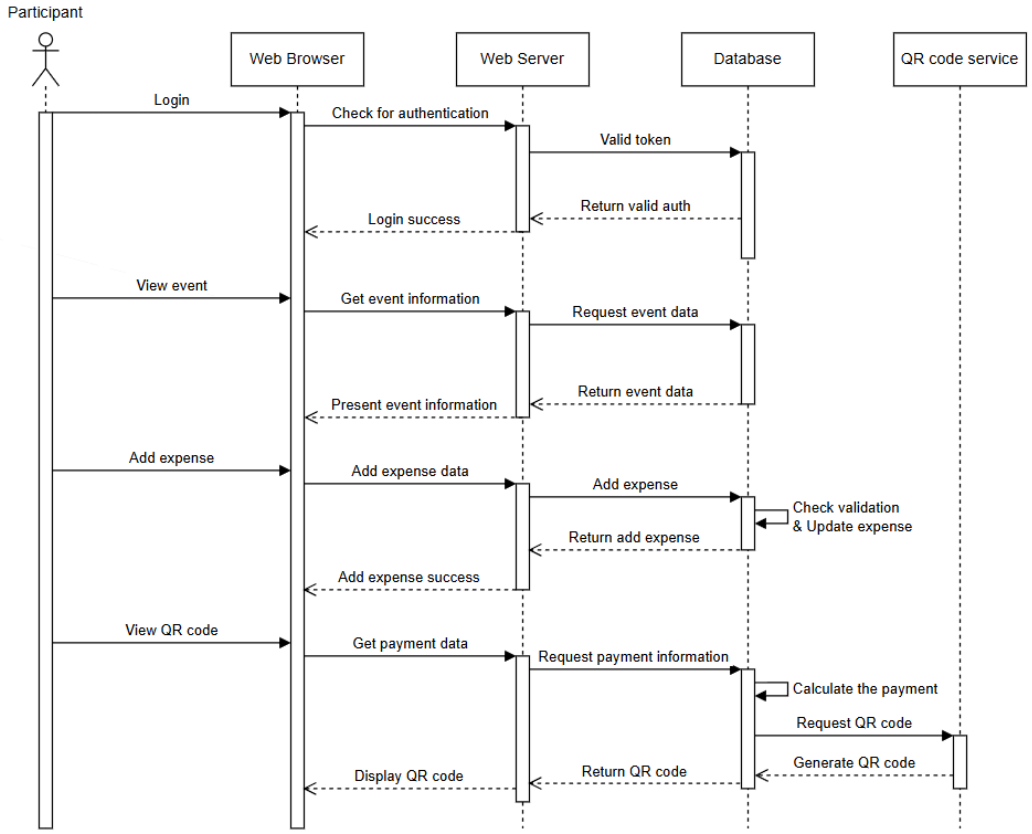

Figure 7: System sequence diagram

The Sequence Diagram for the Share Bill system provides a detailed view of how various system components—such as the User Interface, the Application Server, and the Database—collaborate to execute specific business processes. While Use Case diagrams identify the system's functions, the Sequence Diagram maps out the dynamic behavior by illustrating the chronological order of messages exchanged between objects. This ensures a clear understanding of the logic behind core features like automated cost-splitting and real-time payment updates, serving as a vital bridge between functional requirements and technical implementation.

5 Front-end

5.1 Front-end Overview (Web)

Objective. The Web Front-end provides an interface for users to manage and track expenses within a trip based on an "event", including creating/managing events, managing members, recording transactions (supporting multiple payers and split ratios), viewing expense summaries, and supporting settlement via bank transfer QRs.

Target Users. The system serves two main user groups:

- **Event Organizer:** Creates & manages events, manages participants, manages transactions, calculates & views summary reports, sets up the collector, generates payment QRs, and exports data/closes events.
- **Participant:** Joins events via link, updates information, views event details, contributes transactions, views summaries and personal payment QRs, views transaction history, and leaves events.

Login & Access. Users must log in to access the system; the Login page is the initial interface upon entering the website and contains an Email/Password/Captcha form for authentication.

Main Screens. The Front-end implements screens according to functional groups:

- **Authentication:** Login, Register, Verification Email.
- **Dashboard (Post-login):** Home (event list/navigation), Expenses, Accounts, Settings, etc.
- **Payment Management:** Generate Payment QR Code and related pop-ups (Generated QR, Sharing), supports generating QR for all members or specific individuals, and downloading/sharing QRs.

Backend Interaction. The Front-end calls APIs to perform core functions: creating/editing/deleting events, managing participants, managing transactions, calculating summaries, and generating VietQR based on individual balances.

5.2 Front-end Implementation Flows

Login and Dashboard. Login page is where user enter their account information such as email and password. User fill in the login form and click Login button. After that, frontend send login information to backend to check if the account is valid or not. If login information is correct, system allow user to access dashboard page. If not, system show error message to user.

After user login success, system redirect user to dashboard page. Dashboard page show list of events that user already join. User can see basic information of each event such as event name and date.

Create Event.

In dashboard page, user can create new event by click "Create Event" button. System show a create event form for user to fill event information (event name, date, note...). User click "Create" then frontend send request to backend to create the event. If create success, system redirect to

Activity page and event will appear in event list. If create fail, system show error message and user stay in the form.

Open Activity and View Expenses. After user select an event from dashboard, system open Activity page of that event. Activity page show event name and recent transactions of this activity. At the top, user can see total expenses and balance information such as "you owe" or "you are owed". If there is no transaction yet, system show message "No transactions yet". User can switch between activities by using activity dropdown. User can click Refresh button to reload newest data from backend. User also can invite other members to join this activity by click Invite button.

Add Expense. In Activity page, user click on "New expense" button then system open New expense popup. User fill what is this expense for (example: coffee, ticket, taxi) and enter amount of money. In "Paid by" section, default is user account but user can change to another person by dropdown. User choose participants that this expense is split between. In Split section, user can choose Equal or Exact amounts. If user choose Equal, system split money evenly for selected participants. If user choose Exact amounts, user need to enter exact value for each person. User can add note (optional) then click "Add expense" button to submit. Frontend send expense data to backend to save. If save success, popup will close and new expense will appear in transaction list. Total expenses and balance (you owe / you are owed) will update. If error happen, system show warning and user need check input again.

Settle up and Payment requests. After there are some expenses in an activity, system calculate the balance for each member. Then Settle up section will show who need to pay and who need to receive, with exact amount. User who owe money can click "Pay now" button to start payment. But user only can pay when receiver already add bank information in Accounts page. If receiver have not add bank info yet, system will show message and ask receiver to update bank info first. When bank info is available, system show payment link / QR code for sender to transfer money. After user transfer done, user click "I have paid" to notify the receiver. Then receiver will see a new item in Payment requests section. If receiver already receive money, receiver click "Confirm" to finish this request. If not receive yet, request stay in pending status until receiver confirm later.

5.3 Front-end Technology Stack

The front-end of the system is developed as a web application using modern JavaScript technologies. The main goal is to provide a responsive and easy-to-use interface for users to manage events, expenses, and payment settlement.

Main technologies used in the front-end include:

- **React**: as the main UI framework to build reusable components.
- **Vite**: as the build tool to support fast development and quick reload.
- **TypeScript**: to help reduce common coding errors and make the code easier to understand.
- **Tailwind CSS**: for styling, allowing fast UI development using utility classes.
- **React Query**: to manage server state such as events, expenses, and summary data fetched from backend APIs.

- **Zustand**: to manage global client-side state like authentication status and selected event.

The project follows a feature-based modular architecture. This helps keep the code organized and makes it easier to add new features later.

5.4 Routing and Navigation

The front-end uses client-side routing to control how users navigate between different pages in the application. Each route is mapped to a specific screen such as login, dashboard, event detail, or expense management.

Routing in the system is designed with the following structure:

- **Public routes:**
  - Login, register, and email verification pages.
  - Invite link page that allows users to join an event without logging in first.
- **Private routes:**
  - Main application pages such as event list, activity page, expenses, accounts, and profile.
  - These routes require user authentication before access.
- **Route protection:**
  - If a user tries to access a private route without logging in, the system redirects the user to the login page.
- **Error handling:**
  - A 404 page is displayed when the user accesses an invalid or non-existing route.

This routing approach helps improve both security and user experience.

5.5 Source Code Organization

The front-end source code is organized into multiple modules based on functionality. This structure helps developers understand and maintain the code more easily.

Main organization principles:

- **Page-level components**: handle routing and main screens.
- **Feature modules**: handle business logic such as event management and expense handling.
- **Shared components and utilities**: are reused across the application.
- **Global state**: is separated from local UI state.

This structure supports team collaboration and future system extension.

5.6 Application Initialization

The front-end application is initialized from a single entry point.

During application startup:

- Global providers are set up for authentication state and API data handling.
- Existing authentication state is checked to keep user login session.
- Core application layout and routing are rendered after initialization.

This initialization process ensures consistent behavior across different application pages.

5.7 Authentication Handling

Authentication in the front-end is handled using a token-based mechanism.

Key points include:

- After successful login, the backend returns an access token.
- The token is stored on the client side and attached to subsequent API requests.
- Protected pages require authentication before access.
- When the user logs out, the token is removed and the user is redirected to the Login page.

This approach ensures that only authorized users can access private application features.

5.8 Front-end and Backend Integration

The front-end communicates with the backend system through RESTful APIs to perform all core functionalities of the application.

Main integration points include:

- Sending authentication information during login and receiving access token.
- Creating and managing events and participants.
- Adding expenses and retrieving updated expense data.
- Requesting settlement calculation and generating payment QR codes.

The backend API base URL is configured through environment variables, allowing the front-end to switch between development and production environments easily.

—

5.9 Detailed Features

The front-end system is divided into multiple functional modules. Each module corresponds to a main screen or feature in the application and is responsible for a specific part of the user workflow.

Module	**Main Page/File**	**Description**
Authentication	pages/auth/*	Login, Register, Forgot password functionalities.
Events	events-list-page	Display list of events and allow user to create new event.
Activity	activity-page	Dashboard for managing expenses, settlement process, and payment QR.
Expenses	expenses-page	Expense form handling and debt calculation between members.
Invite Members	invite-page	Generate invite link or QR code to invite new members to an event.
Accounts	accounts-page	User profile management, bank information, and change password.
Settlement	settle-up-modal	Modal to handle settlement and payment flow between members.

6 Backend

6.1 Overview

The database is designed to handle complex expense-splitting scenarios during group trips. It supports multiple payers per expense, custom split ratios for beneficiaries, and a centralized settlement system through a "Collector." The architecture ensures that even if a participant is not a registered system user, their data can still be managed within the scope of a specific event.

6.2 Entity Relationship Diagram and Mapping

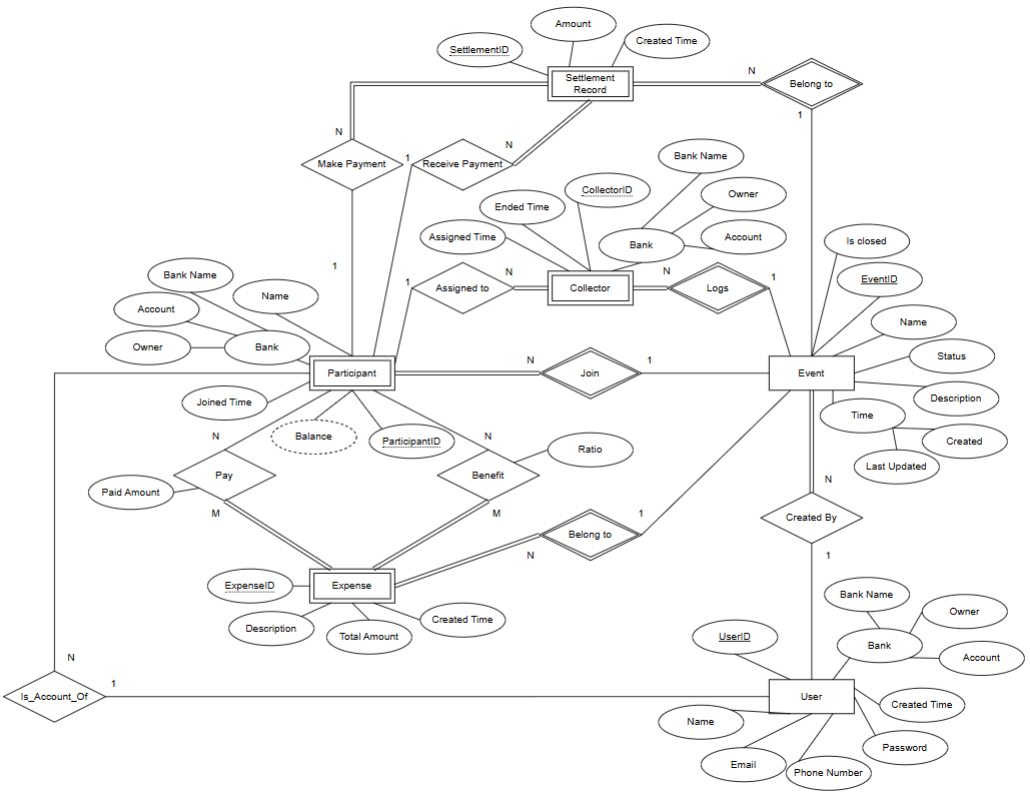

Figure 8: Entity Relationship Diagram

1. USERS & EVENTS: A registered user creates an event. The EVENTS table stores meta-data like status and the unique ID used for the public URL
2. PARTICIPANTS: This is a bridge entity. A participant can be linked to a user_id (if they have an account) or exist only as a name and bank info for that specific event.
3. EXPENSES, PAYERS, & BENEFICIARIES: This is the core of the calculation engine. Instead of a simple one-to-many, we use two many-to-many tables (EXPENSE_PAYERS

and EXPENSE_BENEFICIARIES) to allow multiple people to contribute to a single bill and split the cost using a split_ratio.

4. COLLECTORS & SETTLEMENTS: One or more participants can be assigned as collectors. SETTLEMENT_RECORDS tracks the flow of money between members and the collector to balance the books.

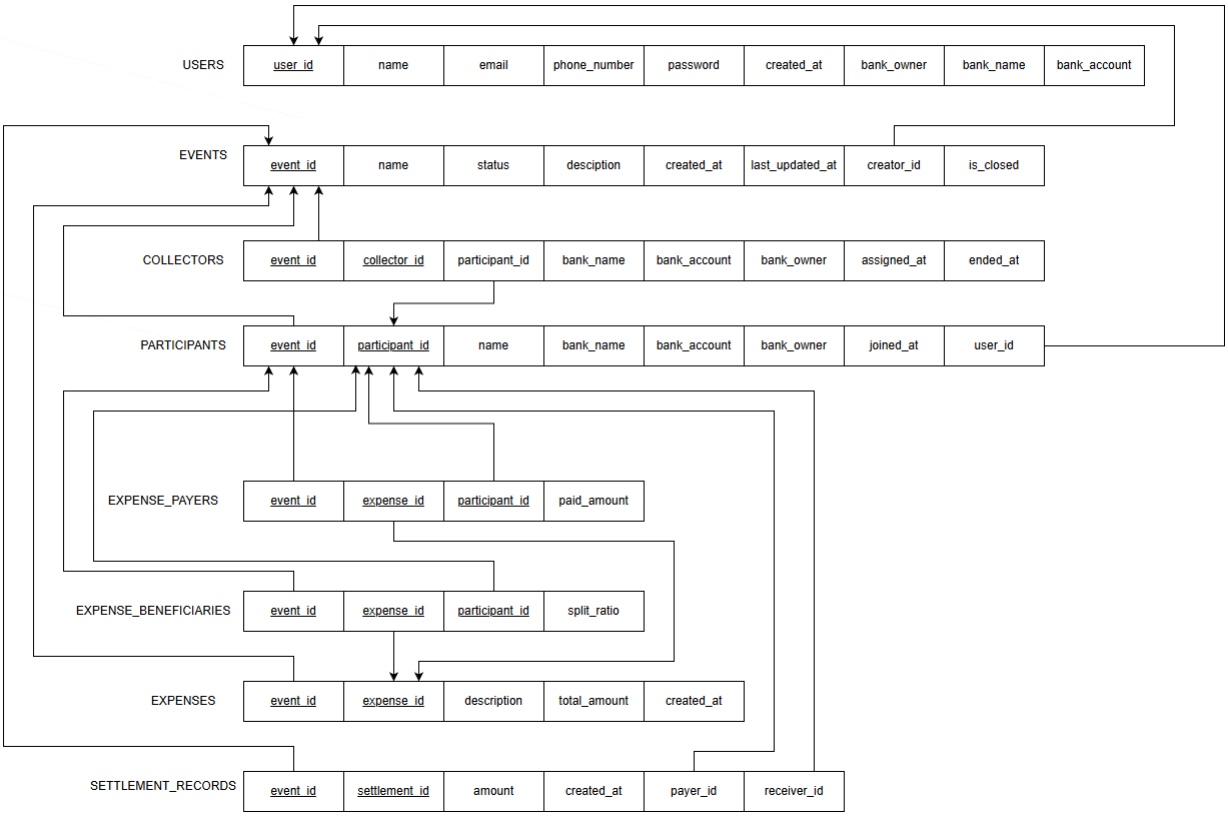

Figure 9: Relational schema

Table 22: Data Dictionary for USERS Table

Attribute	**Data Type**	**Description**
user_id	BIGSERIAL	Primary key, No NULL; Unique internal ID
user_uuid	UUID	Unique external ID for security/API
name	TEXT	No NULL; Display name of the user
email	TEXT	Unique; User's email address
avatar	TEXT	URL to user's profile picture
phone_number	TEXT	Unique; User's contact number
password	TEXT	No NULL; Encrypted password string
bank_name	TEXT	Name of the user's default bank
bank_account	TEXT	User's default bank account number
bank_owner	TEXT	Name of the bank account holder
created_at	TIMESTAMPTZ	Record creation timestamp
updated_at	TIMESTAMPTZ	Last update timestamp

Table 23: Data Dictionary for EVENTS Table

Attribute	**Data Type**	**Description**
event_id	BIGSERIAL	Primary key, No NULL; Internal event ID
event_uuid	UUID	Unique ID used for the public bill URL
name	TEXT	No NULL; Name of the trip/event
status	TEXT	Current status (active, pending, etc.)
description	TEXT	Detailed notes about the event
currency	TEXT	Default currency (e.g., VND, USD)
created_at	TIMESTAMPTZ	Time when the event was created
last_updated_at	TIMESTAMPTZ	Time of the last record modification
creator_id	BIGINT	Foreign key; Links to users(user_id)
is_closed	BOOLEAN	No NULL; Whether the event is finalized

Table 24: Data Dictionary for PARTICIPANTS Table

Attribute	Data Type	Description
participant_id	BIGSERIAL	Primary key, No NULL; Participant ID
participant_uuid	UUID	Unique external ID for the participant
event_id	BIGINT	Foreign key; Links to events(event_id)
user_id	BIGINT	Foreign key; Links to users(user_id) or NULL
name	TEXT	No NULL; Participant's name in this event
bank_name	TEXT	Bank name specific to this event
bank_account	TEXT	Account number for this event
bank_owner	TEXT	Account owner name for this event
joined_at	TIMESTAMPTZ	Time the person joined the event

Table 25: Data Dictionary for EXPENSES Table

Attribute	Data Type	Description
expense_id	BIGSERIAL	Primary key; Unique bill ID
expense_uuid	UUID	Unique external ID for the expense
event_id	BIGINT	Foreign key; Links to events(event_id)
description	TEXT	No NULL; Purpose of the expense
total_amount	NUMERIC	No NULL; Total cost of the bill
created_at	TIMESTAMPTZ	Time when the expense was recorded

Table 26: Data Dictionary for EXPENSE_PAYERS Table

Attribute	Data Type	Description
payer_id	BIGSERIAL	Primary key; Record of payment
payer_uuid	UUID	Unique external ID for the payer record
expense_id	BIGINT	Foreign key; Links to expenses(expense_id)
participant_id	BIGINT	The person who actually paid money
paid_amount	NUMERIC	The amount contributed by this person

Table 27: Data Dictionary for EXPENSE_BENEFICIARIES Table

Attribute	**Data Type**	**Description**
beneficiary_id	BIGSERIAL	Primary key; Record of consumption
beneficiary_uuid	UUID	Unique external ID for the beneficiary
expense_id	BIGINT	Foreign key; Links to expenses(expense_id)
participant_id	BIGINT	The person who benefits from the expense
split_ratio	NUMERIC	Weight/Ratio for splitting the cost

Table 28: Data Dictionary for SETTLEMENTS Table

Attribute	**Data Type**	**Description**
settlement_id	BIGSERIAL	Primary key; Settlement record ID
settlement_uuid	UUID	Unique external ID for the settlement
event_id	BIGINT	Foreign key; Links to events(event_id)
amount	NUMERIC	The final amount transferred
payer_id	BIGINT	Participant sending the money
receiver_id	BIGINT	Participant receiving the money

6.3 Implementation

6.3.1 Data Definition Language

```
CREATE TABLE IF NOT EXISTS users(
    user_id BIGSERIAL PRIMARY KEY,
    user_uuid uuid NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    name TEXT NOT NULL,
    email TEXT UNIQUE,
    avatar TEXT,
    phone_number TEXT UNIQUE,
    password TEXT NOT NULL,
    bank_name TEXT,
    bank_account TEXT,
    bank_owner TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS events(
    event_id BIGSERIAL PRIMARY KEY,
    event_uuid UUID NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    name TEXT NOT NULL,
    status TEXT DEFAULT 'active',
    description TEXT,
    currency TEXT NOT NULL DEFAULT 'VND',
    created_at TIMESTAMPTZ DEFAULT now(),
    last_updated_at TIMESTAMPTZ DEFAULT now(),
    creator_id BIGINT REFERENCES users(user_id),
    is_closed BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE IF NOT EXISTS participants(
    participant_id BIGSERIAL PRIMARY KEY,
    participant_uuid UUID  NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    event_id BIGINT NOT NULL REFERENCES events(event_id) ON DELETE CASCADE
    ,
    user_id BIGINT REFERENCES users(user_id) ON DELETE SET NULL,
    name TEXT NOT NULL,
    bank_name TEXT,
    bank_account TEXT,
    bank_owner TEXT,
    joined_at TIMESTAMPTZ DEFAULT now(),
    UNIQUE(event_id, user_id)
);

CREATE TABLE IF NOT EXISTS collectors(
    collector_id BIGSERIAL PRIMARY KEY,
    collector_uuid UUID NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    event_id BIGINT NOT NULL REFERENCES events(event_id) ON DELETE CASCADE
    ,
    participant_id BIGINT REFERENCES participants(participant_id) ON
   DELETE SET NULL,
    bank_name TEXT NOT NULL,
    bank_account TEXT NOT NULL,
    bank_owner TEXT NOT NULL,
```

```
    assigned_at TIMESTAMPTZ DEFAULT now(),
    ended_at TIMESTAMPTZ,
    is_active BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE IF NOT EXISTS expenses (
    expense_id BIGSERIAL PRIMARY KEY,
    expense_uuid UUID NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    event_id BIGINT NOT NULL REFERENCES events(event_id) ON DELETE CASCADE,
    description TEXT NOT NULL,
    total_amount NUMERIC(14,2) NOT NULL CHECK (total_amount >= 0),
    created_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE IF NOT EXISTS expense_payers (
    payer_id BIGSERIAL PRIMARY KEY,
    payer_uuid UUID NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    expense_id BIGINT NOT NULL REFERENCES expenses(expense_id) ON DELETE CASCADE,
    participant_id BIGINT REFERENCES participants(participant_id) ON DELETE SET NULL,
    paid_amount NUMERIC(14,2) NOT NULL CHECK (paid_amount >= 0)
);

CREATE TABLE IF NOT EXISTS expense_beneficiaries (
    beneficiary_id BIGSERIAL PRIMARY KEY,
    beneficiary_uuid  UUID UNIQUE NOT NULL DEFAULT gen_random_uuid(),
    expense_id BIGINT REFERENCES expenses(expense_id) ON DELETE CASCADE,
    participant_id BIGINT REFERENCES participants(participant_id) ON DELETE SET NULL,
    split_ratio NUMERIC(10,4) NOT NULL
);

CREATE TABLE IF NOT EXISTS settlements (
    settlement_id BIGSERIAL PRIMARY KEY,
    settlement_uuid UUID NOT NULL UNIQUE DEFAULT gen_random_uuid(),
    event_id BIGINT NOT NULL REFERENCES events(event_id) ON DELETE CASCADE,
    amount NUMERIC(14,2) NOT NULL CHECK (amount >= 0),
    created_at TIMESTAMPTZ DEFAULT now(),
    payer_id BIGINT REFERENCES participants(participant_id),
    receiver_id BIGINT REFERENCES participants(participant_id)
);
```

Listing 1: Create Schema

6.3.2 Database Functions and Triggers

To ensure data integrity and optimize read performance, the system utilizes PostgreSQL Trigger Functions to maintain denormalized summary columns in the events table.

update_event_participant_stats: This trigger function maintains real-time consistency for the total_participants counter in the events table. It automatically increments the counter

upon the insertion of a new participant and decrements it when a participant is removed, ensuring the summary data remains accurate without manual recalculation.

```sql
CREATE OR REPLACE FUNCTION update_event_participant_stats()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        UPDATE events SET total_participants = total_participants + 1
    WHERE event_id = NEW.event_id;
    ELSIF (TG_OP = 'DELETE') THEN
        UPDATE events SET total_participants = total_participants - 1
    WHERE event_id = OLD.event_id;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_participant_stats
AFTER INSERT OR DELETE ON participants
FOR EACH ROW EXECUTE FUNCTION update_event_participant_stats();
```

Listing 2: update event participant stats function and trigger

update_event_expense_stats: This function serves as a financial synchronization engine. It monitors the expenses table and automatically updates the parent events table. It handles INSERT by adding amounts, DELETE by subtracting them, and UPDATE by recalculating the difference between old and new values, maintaining perfect data integrity for the event's total balance and transaction count.

```sql
CREATE OR REPLACE FUNCTION update_event_expense_stats()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        UPDATE events
        SET total_transactions = total_transactions + 1,
            total_expenses = total_expenses + NEW.total_amount
        WHERE event_id = NEW.event_id;

    ELSIF (TG_OP = 'DELETE') THEN
        UPDATE events
        SET total_transactions = total_transactions - 1,
            total_expenses = total_expenses - OLD.total_amount
        WHERE event_id = OLD.event_id;

    ELSIF (TG_OP = 'UPDATE') THEN
        IF OLD.total_amount <> NEW.total_amount THEN
            UPDATE events
            SET total_expenses = total_expenses - OLD.total_amount + NEW.
    total_amount
            WHERE event_id = NEW.event_id;
        END IF;
    END IF;
    RETURN NULL;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_expense_stats
AFTER INSERT OR UPDATE OR DELETE ON expenses
FOR EACH ROW EXECUTE FUNCTION update_event_expense_stats();
```

Listing 3: update event expense stats function and trigger