Splitty Backlog

Stakeholder

- User
- Admin

Terminology

Domain:

- **Event:** Any event that involved a group of people, which have shared expenses
- **Participant:** A person that joined a group event
- **Expense:** An item or activity that has been paid by one participant, for at least one other participant.
- **Debt:** An amount of money one person owes another person. In a group, debts can be *collective* (e.g., two participants need to pay a third participant for their share of a taxi ride) or *partial* (e..g, Anna paid drinks for herself and for Ben, Ben paid for the food. Assuming the food was more expensive, Anna only owes Ben half of the difference between drinks and food).

Client Application:

- **Starting Page:** The starting page is the overview over all events that is shown when the application is started. An example is given in Mock 1.
- **Event Overview:** Once an event has been created/selected, the application will show an overview over the details of one specific event (see Mock 2).
- **Client Config:** The client has several config parameters that should not be hardcoded. As such, the application has a config file, which makes these parameters adaptable.
- **Session:** The period in which a single client instance is running.

Non-Functional Requirements

- The application is built with Spring/JavaFX
- Client/Server Communication is performed via HTTP and JSON through REST and websockets.
- The system needs to illustrate the use of long-polling via REST and websockets. The majority of the communication can be implemented in one style, but there must be at least one example of each to illustrate that all learning objectives of the course have been reached.
- The application should not require the implementation of a user account system
- It must be possible to connect at least 10 clients simultaneously
- Change propagation between clients feels “instant” (takes less than 1s)

Epics and User Stories

Basic Requirements

Note that all(!) basic requirements must be met to pass the course.

The basic requirements cover a basic expense calculator. The application has a client-server architecture. While it must be possible to connect multiple clients simultaneously to one server, the client implementation is still very basic and does not contain many of the advanced use cases. The main purpose is to create an extensible platform as a basis for the remaining tasks, which allows the teams to apply the various mandatory technologies.

As a user, I want…

- To connect to a Splitty server of my choice without recompilation, so I can manage expenses
  (The server url should be added to a config file, to avoid a hard-coding and make it changeable)
- To connect multiple clients at once to one server, so two users can use Splitty simultaneously
- To create or join an event, so I can manage expenses for that event
- To see an invite code in the overview of an event that I can give to others, so they can join
- To add/edit/remove participants to an event, so I can create an complete and accurate list
- To give an event a title, so it is easy to differentiate two events
- To edit the title of an event, so I can fix typos
- To see all created expenses for the current event, so I see what has already been entered
- To add new expenses to the event, so I can add missing items or activities.
  (In this basic form, an expense only requires a paying participant, an amount, and a title, expenses are assumed to be split equally among all participants)
- To edit or remove existing expenses, so I can correct mistakes
- To see the total sum of all expenses, the share per person and how much each person owes to/is owed by the group, so all participants can meet and settle their debts.
  (This is the bare minimum to settle the debt and inconvenient. It will be extended later)
- To switch between events, so I can manage expenses of multiple events in the same session
- To switch between English and Dutch, so I can use my preferred language.
  (In this basic form, the language should be configured in the config file and must only be considered once when starting of the application)

As an admin, I want…

- To login into a password-protected management overview, so I can manage my server instance
- To see a random password in the server output, so I can login to the management overview.
- To see all events of the server, so I can explore the existing data.
- To order all existing events by title, creation date, and last activity, so I can find specific events
- To delete events, so I can clean up the database
- To download a JSON dump of a selected event, including all details, so I can create backups
- To import an event from a JSON dump, so I can restore backups.
  (the handling of repeated imports or import conflicts remains undefined)

Live Language Switch

Add support for switching the language of your client during runtime. The language selection should be persisted in the config file.

As a user, I want…

- To see a language indicator on the starting page and the event overview, so I know which language is currently configured in my client on a first glance
- To see a flag icon as the language indicator, so I do not have to read additional text.
- To see all available languages through clicking the indicator, so I can find my preferred one easily
- To persist my language choice through restarts, so I do not have to pick a language each time
- To generate an empty language template from the language selection, so I can contribute a new language to the project.

Non-functional Requirements:

- The application must fully support at least English and Dutch. A third language must be added as a proof of concept, but can be made up. Please note that languages that do not flow left-to-right are much harder to integrate. While we encourage you to think about this use case, it is enough to limit your localization support to left-to-right languages.

Detailed Expenses

The purpose of this extension is to support more complex payment situations and add appropriate information to the overview. The main use cases covered are 1) person A giving money to person B and 2) expenses that do not cover all participants (e.g., a subset of participants are sharing a cab).

As a user, I want…

- To give expenses a date, so I can find them easier, for example, in my bank account
- To decide for expenses whether to split them “equally” or only in a “subgroup”, so I can select those participants that were involved in the expense.
- To add support for giving money from A to B, so I can represent a basic exchange of money during an event or a partial settling of debt.
- To see all registered expenses in an overview, so I can get a grasp of the total money spent.
- To filter the overall expense overview, so I can explore all expenses more easily.
  (It is not necessary to support generic filter, it is ok, if only the following filters get implemented)
- To filter the list to only my expenses, so I can check easily that I did not forget something.
- To filter to all expenses that involve me, so I can easily double-check for what I have to pay.
- To see for each expense the date, who paid, how much, and the involved participants, so I can understand the details.

Foreign Currency

Add support for expenses in foreign currencies. The relevant exchange rates should be automatically fetched from an external web service*.

As a user, I want…

- To configure my preferred currency, so I can pick the one that matches my mental model (This preference should be persisted in the configuration file)
- To specify a currency for an expense, so I can manage expenses in multiple currencies
- To specify the date for an expense, so its amount can be converted
- To see only money values in my preferred currency, so I can better understand the amounts
- To have the system automatically fetch exchange rates, so I can avoid manual maintenance

Non-Functional Requirements:

- The application must work with at least USD, EUR, and CHF
- Only the server will communicate with the exchange rate service. It will provide its own GET service, which will allow all connected Splitty clients to request conversion rates or conversions. (It is up to the group to decide whether the backend service will consume an input “amount” and produce its converted value or whether the service will only consume and produce rates.)
- Requested exchange rates will be cached in a local file (e.g., rates/«date»/«from»/«to».txt), repeated requests with the same parameters will reuse the previously fetched exchange rates.

(*) You can find free exchange rate converters online, which allow conversions via REST API, like:

https://fixer.io/
https://openexchangerates.org/
https://free.currencyconverterapi.com/

These services are free, but usually require a registration and the creation of an API key. Even on their free tiers, the available rate limit will be much higher than what should be required for the project.

Should a team not want to register with such a third party, they are allowed to create their own “fake” exchange converter. A basic GET service can calculate a pseudo random conversion rate for a given date and exchange pair, for example, a valid recent rate, with a random +-10% adjustment (recommendation: use the timestamp as a seed for the randomization to create stable rates and make sure that “rate(A,B)=1/rate(B,A)”). Important: There must be a separate endpoint and an explicit REST call to this endpoint, as if the requesting code would use one of the real services listed before.

Open Debts

The application calculates the debt per person and displays concrete payment instructions to settle the open debt within the group.

As a user, I want…

- To get payment instructions, so I can settle my debts
  (The system should calculate the (at most) N-1 transfer instructions that are required in a group of N people; the total amount of transferred money should be correct and minimal)
- To mark when I have received the money from someone, so I can mark a debt as settled.
  (Hint: To avoid special handling, represent this as a “Give money from A to B” expense)
- To only see those participants in the open debt list, who still owe money, so I can find a name more easily
- To add bank accounts to participants, so the payment instructions can involve a bank transfer.
  (Bank accounts should remain optional for each participant)
- To have each payment instruction consist of a short summary and expandable/collapsable details, so I can hide the details, when I do not need them.

Statistics

This extension will introduce a tag system for the expenses and calculate basic statistics about the expenses. The tags are a cross-cutting concern and will also affect other parts of the application, like the overview.

As a user, I want…

- To select a tag from a list when creating an expense, so I can organize expenses by type. (The selection drop down does not need to be colored)
- To have the three standard tags food, entrance fees, travel for each new event, so I can classify the expenses.
- To add new tags, so I can define my own expense types.
- To have a statistics page, so I can explore the distribution of expenses for an event.
- To see the total expense amount for the whole event on the statistics page, so I know how expensive the whole event was.
- To see a basic pie chart that provides an overview over the expenses per tag, so I can understand how the different types of expenses contributed to the overall cost.
- To see the tag of an expense in the expense overview, so I can see at first glance, if all expenses are correctly tagged.
- To define or change the color of a tag, so I can use my preferred color coding.
- To rename or delete tags, so I can pick tags that are most useful for my event.

Non-functional Requirements:

- In the application, tags are always displayed with a colored background.
- The pie chart contains absolute and relative values.

Email Notification

This extension will end the ability to send automated notification and reminders to the participants, conveniently from within the app. This feature depends on the availability of the payment instructions and it should be implemented at the last feature of the project.

As a user, I want…

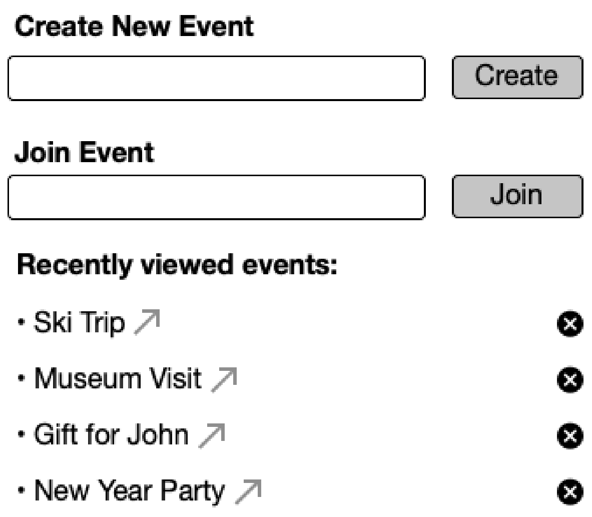
- To configure my email credentials, so I can send notifications or reminders.
  (There is no need for a UI, the credentials can be added to the config file)
- To initiate the sending of a default email, so I can test whether my credentials are correct and the email is delivered.
  (The email has a default body and is being sent to your own email address)
- To send invitation emails to new participants, so I can easily invite new participants to the event.
  (The invitation email must contain the server url and the invitation code that is required to join)
- To automatically add invited participants to the event, so I do not have to manually add them.
- To add/change/remove email addresses in the details of each participant, so I can fix typos.
- To send payment invitations via button click in the debt overview, so I do not have to manually contact all participants.

Non-functional Requirements:

- The application must be usable without or with invalid email credentials.
- Email related features should be disabled/grayed out when a) no credentials are available or b) the target participant has no known email address
- The email configuration must only be stored on the client.
- The email configuration contains your own email address.
- All sent emails are sent from your email address and include your email address in CC.

Mock-Ups

Start Screen


Create New Event

Create

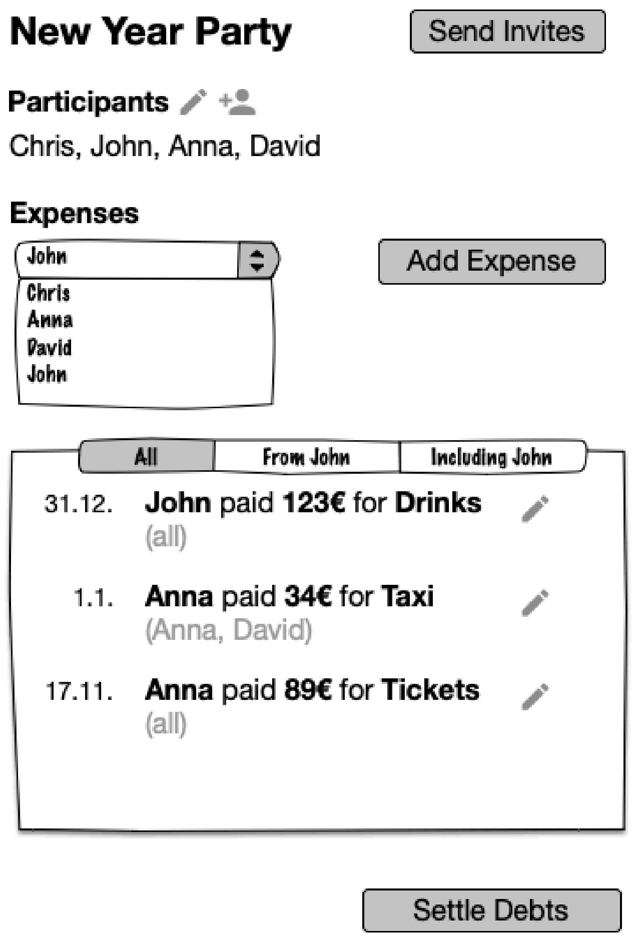
Join Event

Join

Recently viewed events:

- Ski Trip
- Museum Visit
- Gift for John
- New Year Party

Overview


New Year Party

Send Invites

Participants

Chris, John, Anna, David

Expenses

John

Chris
Anna
David
John

Add Expense

All | From John | Including John

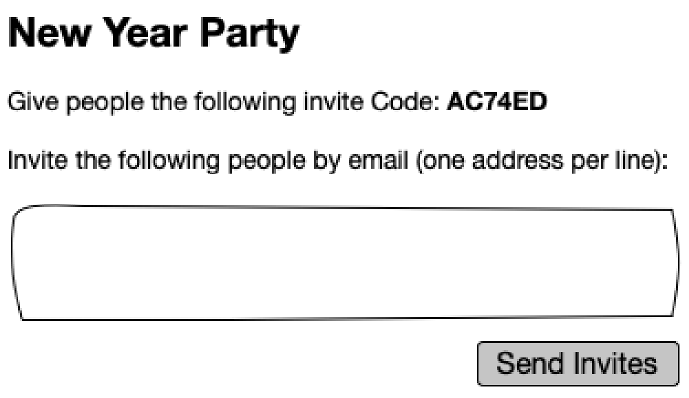
31.12. **John** paid **123€** for **Drinks**
(all)

1.1. **Anna** paid **34€** for **Taxi**
(Anna, David)

17.11. **Anna** paid **89€** for **Tickets**
(all)

Settle Debts

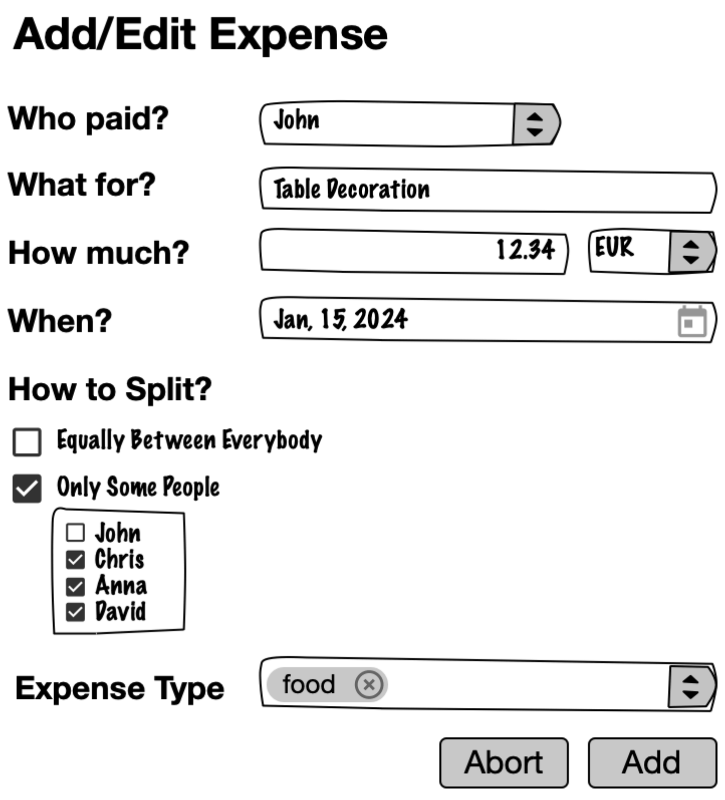
Invitation


New Year Party

Give people the following invite Code: **AC74ED**

Invite the following people by email (one address per line):

Add Expense


Add/Edit Expense

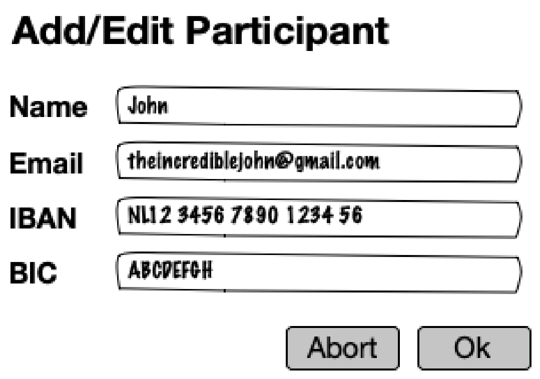
Who paid?	John
What for?	Table Decoration
How much?	12.34 EUR
When?	Jan, 15, 2024

How to Split?

- [ ] Equally Between Everybody
- [x] Only Some People
  - [ ] John
  - [x] Chris
  - [x] Anna
  - [x] David

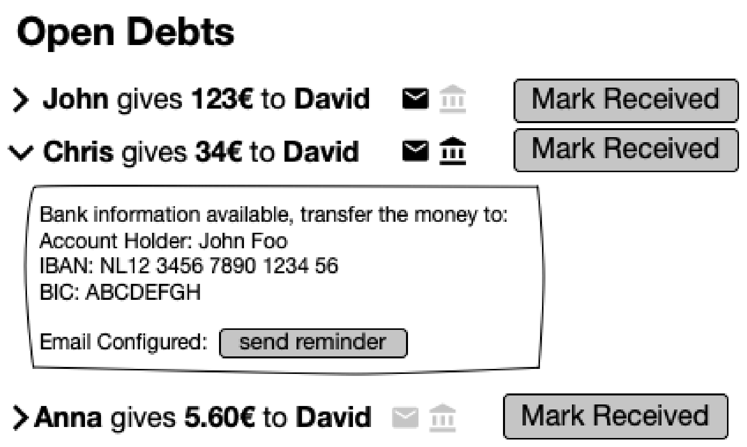
Expense Type food

Contact Details


Add/Edit Participant

Name	John
Email	theincrediblejohn@gmail.com
IBAN	NL12 3456 7890 1234 56
BIC	ABCDEFGH

Open Debts



Open Debts

John gives 123€ to David — Mark Received

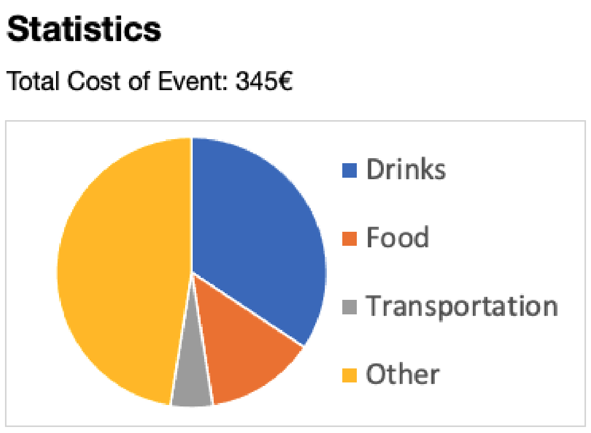
Chris gives 34€ to David — Mark Received

Bank information available, transfer the money to:
Account Holder: John Foo
IBAN: NL12 3456 7890 1234 56
BIC: ABCDEFGH

Email Configured: send reminder

Anna gives 5.60€ to David — Mark Received

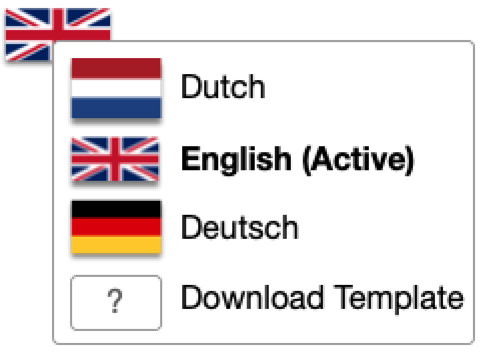
Statistics



Statistics

Total Cost of Event: 345€

Language Selection


Dutch
English (Active)
Deutsch
?
Download Template