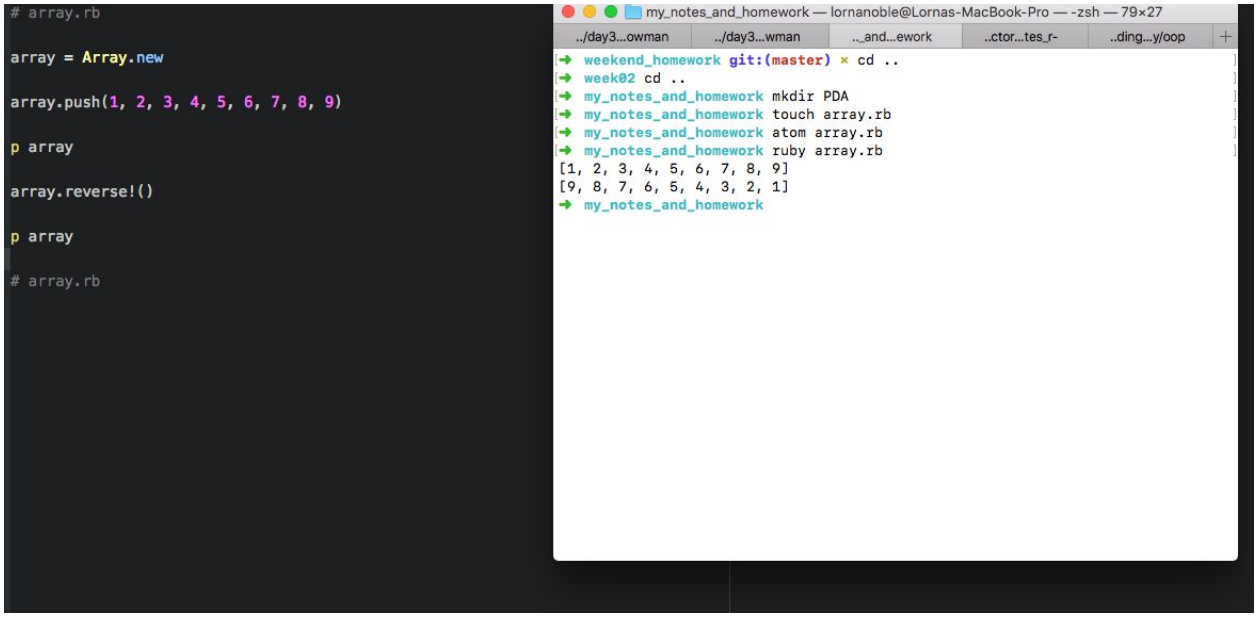
Demonstrate the use of an Array in a program:
An array
A function using the array
The result of the function running

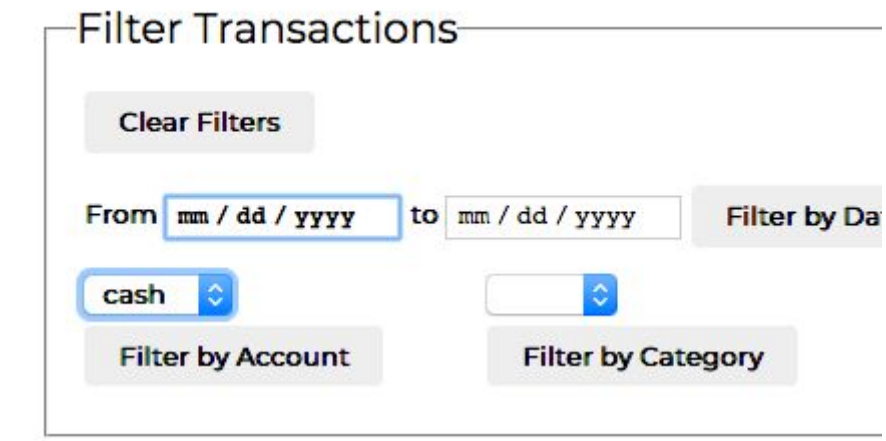


Demonstrate the use of a Hash in a program:
The Hash
A Function using hash
The Result of the function

Demonstrate searching data in a program:

accoun...×

```ruby
def transactions()
  sql = "SELECT * FROM transactions WHERE account_id = $1;"
  values = [@id]
  sql_result = SqlRunner.run(sql, values)
  transactions = sql_result.map {|hash| Transaction.new(hash)}
end
```



Filtered Transactions

Date	Account	Vendor	Category	Amount
22/12/2017	cash	wonder world	fun	£ 3.25
Total:				£ 3.25

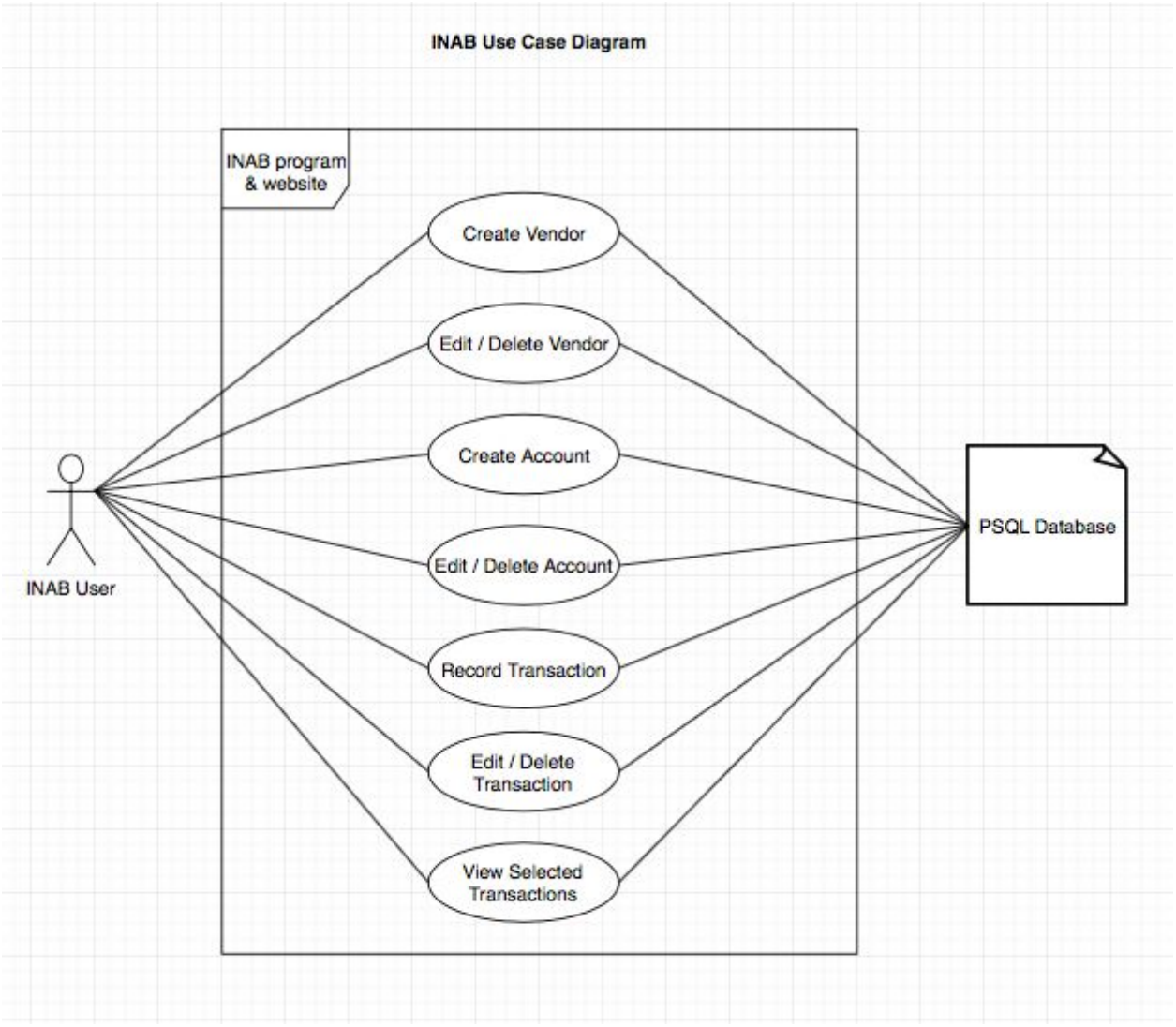
Demonstrate sorting data in a program

```
def self.select_by_account(account_id)
  sql = "SELECT * FROM transactions WHERE account_id = $1 ORDER BY
   transaction_date DESC;"
  values = [account_id]
  sql_result = SqlRunner.run(sql, values)
  transactions_array = sql_result.map {|hash| Transaction.new(hash)}
end
```

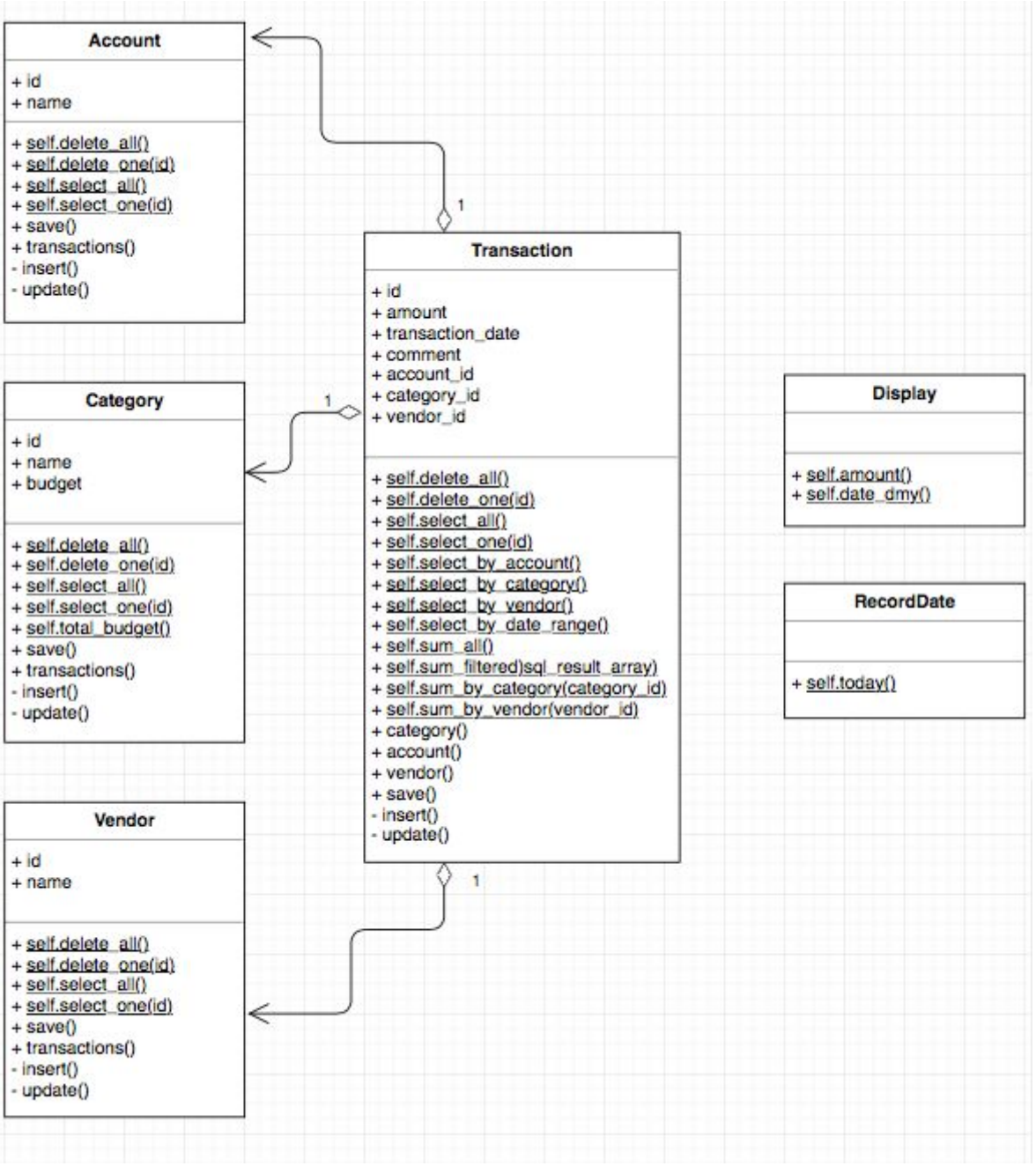
Filtered Transactions

Date	Account	Vendor	Category	Amount
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 8.00
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 12.00
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 94.00
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 10.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 2.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 10.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 174.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 6.00
Total:				£ 316.00

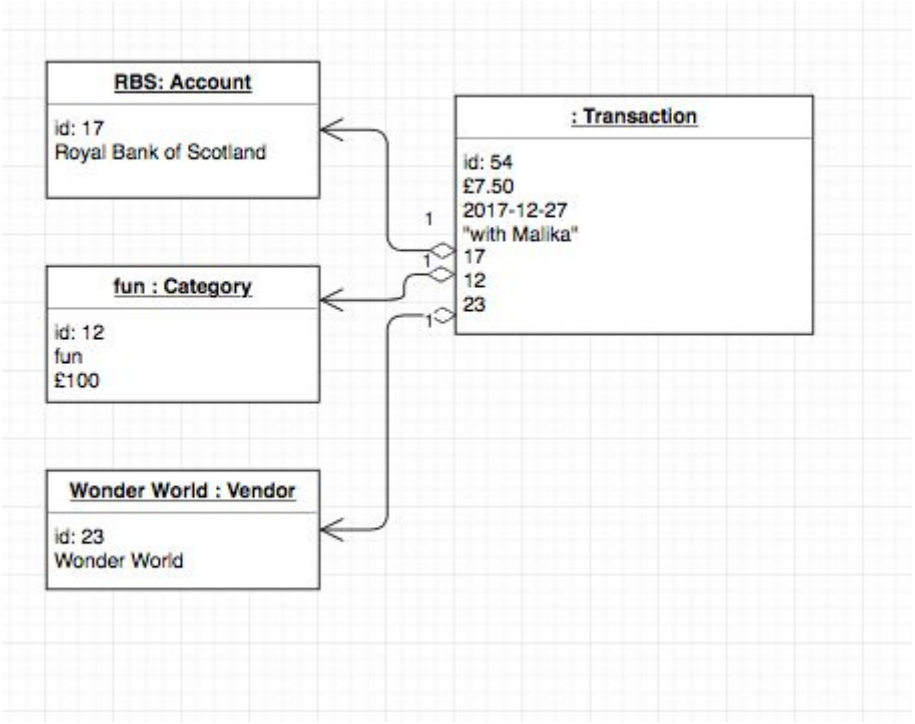
A Use Case Diagram


INAB Use Case Diagram
INAB program & website
Create Vendor
Edit / Delete Vendor
Create Account
Edit / Delete Account
Record Transaction
Edit / Delete Transaction
View Selected Transactions
INAB User
PSQL Database

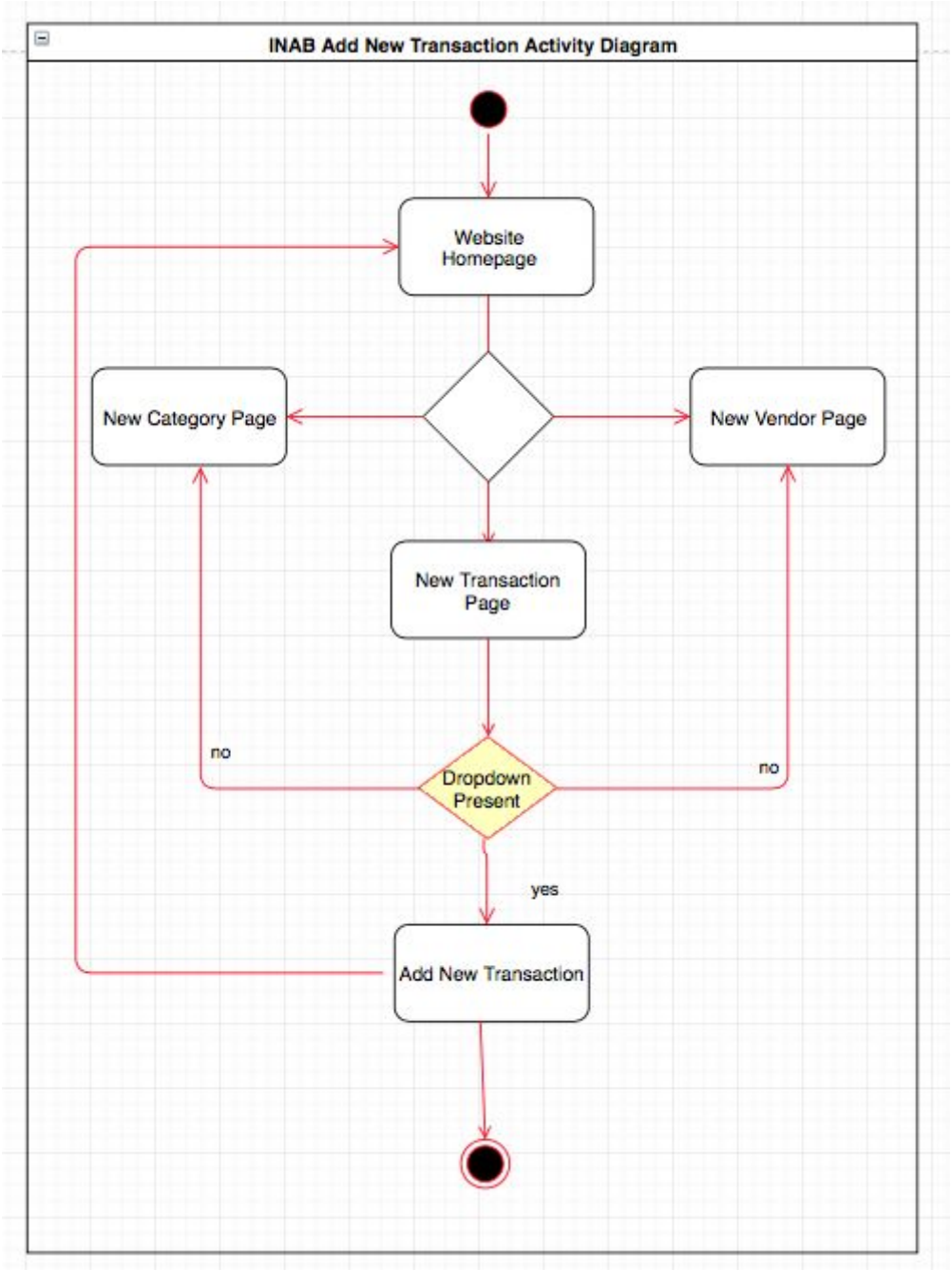
A Class Diagram



An Object Diagram



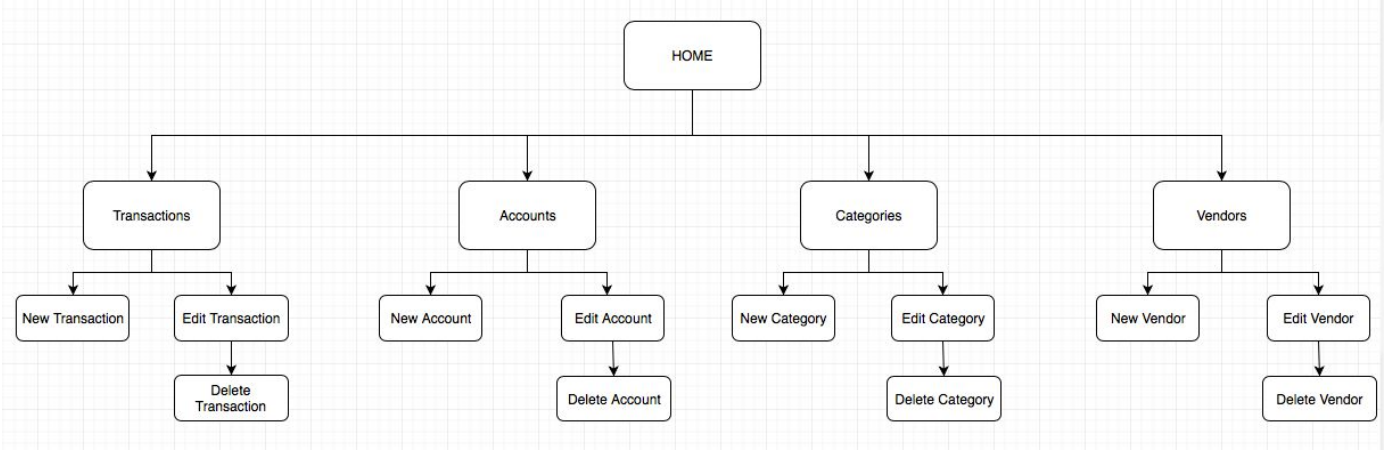
An Activity Diagram



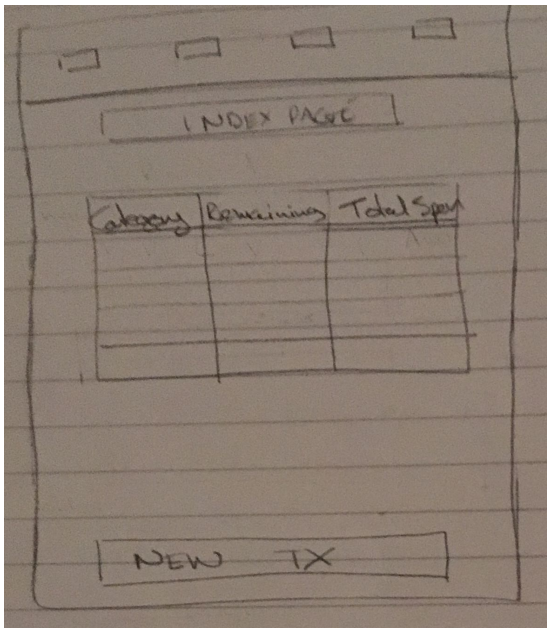
An Implementations Constraints Plan:

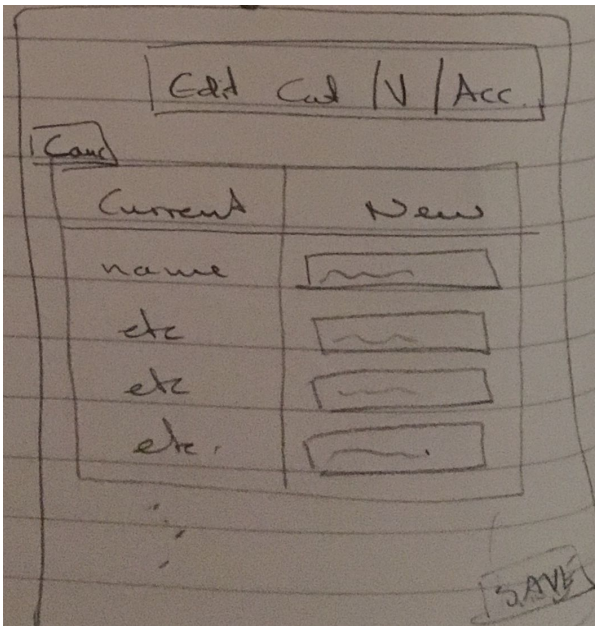
Implementations Constraints Plan		
Constraint	Possible effect on product	Solution
Hardware and software platforms: Development will be carried out on a 2011 MacBook Pro using Ruby, Sinatra, PostgresSQL, HTML and CSS.	Product may not be optimised for viewing on mobile devices, limiting usability.	Use dev tools to view site on different size screens.
Performance requirements: Product is required to run on the Chrome web browser: no specified speed or other performance criteria.	There may be some cross-browser display differences. The site may not function well with slow internet connections.	Accept display differences as insignificant. Test response times using dev tools to model different data rates.
Persistent storage and transactions: Data will be maintained on a PostgresSQL database, with a single user at any one time.	No requirement for user login process or secure data. Limited real-world utility as each user would need to set up their own instance of the program.	Accept that this is a constraint of the project. Consider the need for database transactions to be complete if multi-user setup was instigated.
Usability: Product is required to run on the Chrome web. No access to screen-reader for testing.	There may be some cross-browser display differences. Product may not be fully accessible using assistive technology	Ensure all menus and forms are tab- and keyboard-accessible as well as mouse-activated. Use semantic html for maximum screen reader compatibility.
Budgets: No additional funds are available	Product will be necessarily simple and proof-of-concept.	Accept that this is a constraint of the project.
Time: Seven days, including two non-working weekend days.	Product will be necessarily simple and proof-of-concept.	Accept that this is a constraint of the project.

Create a user sitemap



Produce two wireframe designs



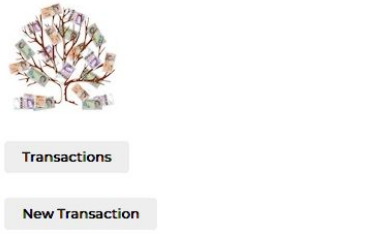


Take a screenshot of an example of pseudocode for a function

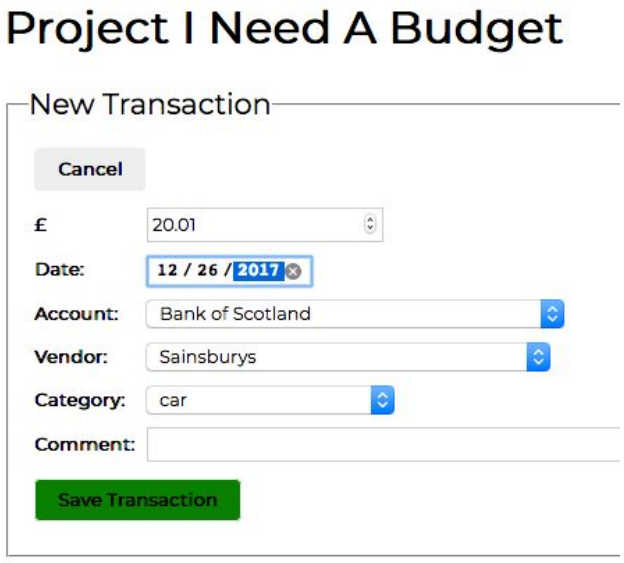
```
def Ticket.sell(customer, screening)
  if customer.can_afford_screening_price?
    if screening.has_seats_empty?
      ticket = Ticket.new(customer, film, screening)
      customer_wallet -= ticket_price
      screening_empty_seats -= ticket_count
      cinema_till += ticket_price
    end
  end
end
```

Show user input being processed according to design requirements:

1. User input

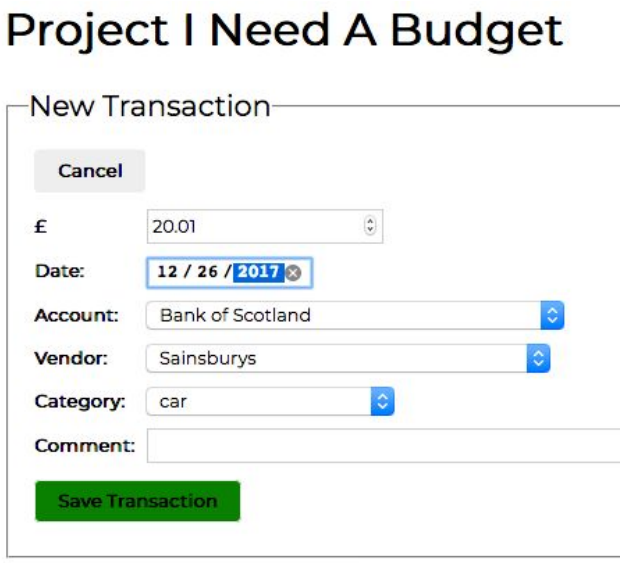


2. User input being saved/used in some way (Sainsbury's vendor used in new transaction)



Show an interaction with data persistence:

1. Data being inputted into your program



2. Confirmation of the data being saved

All Transactions

Date	Account	Vendor	Category	Amount
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 8.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 6.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 174.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 10.00
27/12/2017	Royal Bank of Scotland	Sainsburys	car	£ 2.00
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 10.00
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 94.00
27/12/2017	Royal Bank of Scotland	Waitrose	gifts	£ 12.00
26/12/2017	Bank of Scotland	Sainsburys	car	£ 20.01
7/12/2017	Bank of Scotland	Jemma	lending	£ 0.75
7/12/2017	Bank of Scotland	Jemma	lending	£ 21.75
7/12/2017	Bank of Scotland	Jemma	lending	£ 1.25
7/12/2017	Bank of Scotland	Jemma	lending	£ 0.25
5/12/2017	Bank of Scotland	Aldi	fun	£ 3.00
5/12/2017	Bank of Scotland	Aldi	fun	£ 87.00
5/12/2017	Bank of Scotland	Aldi	fun	£ 5.00
5/12/2017	Bank of Scotland	Aldi	fun	£ 1.00
Total:				£ 456.01

Show the correct output of results and feedback to a user:

1. User requesting information or an action to be performed
2. User request being processed correctly and demonstrated in the program