

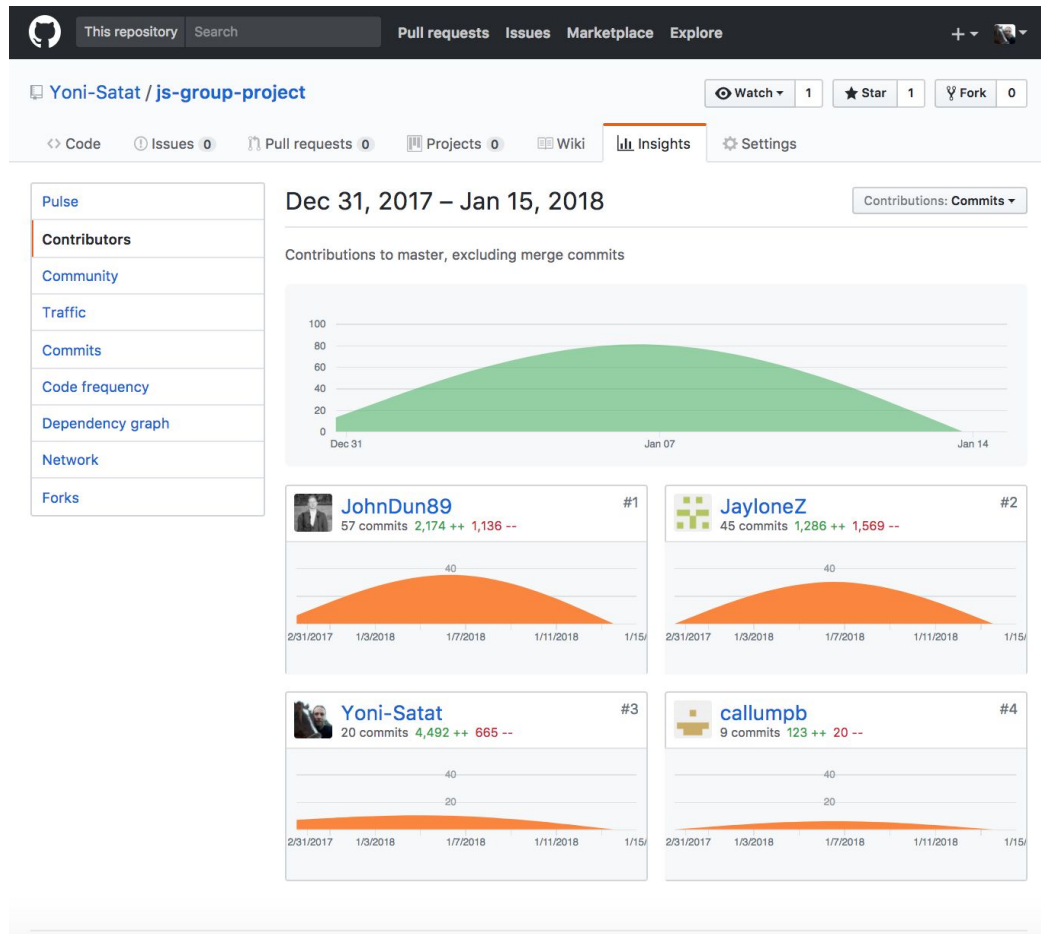
Evidence for Project Unit

Yonatan Satat

E16

25/01/2018

P. 1 Github Contribution Page



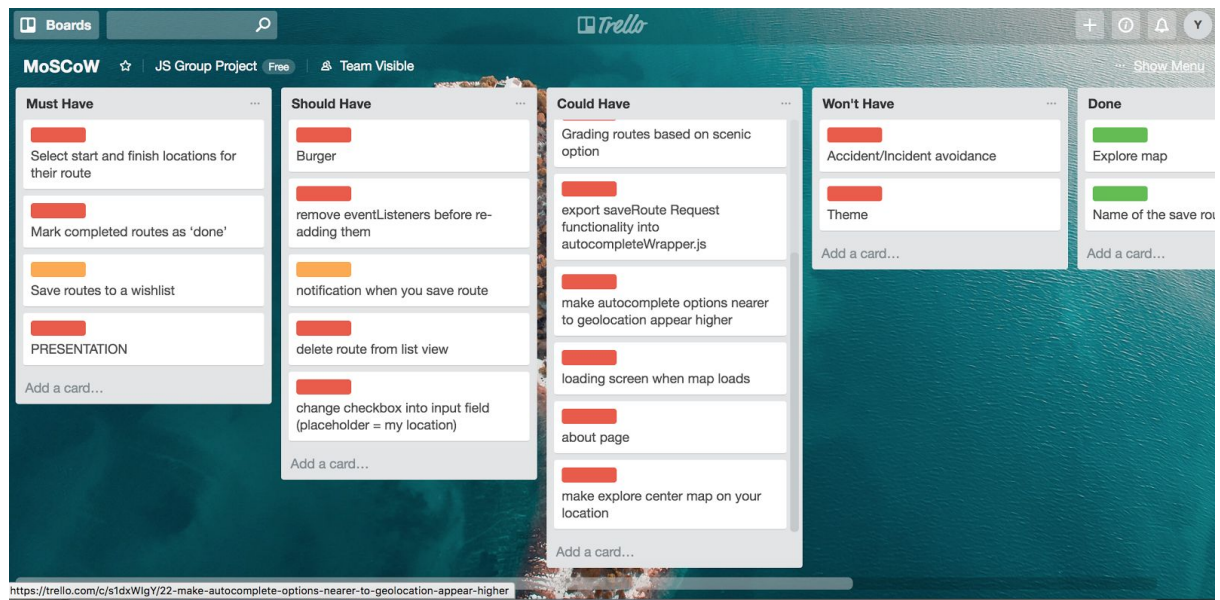
P. 2 Project Brief

Create an app that allows users to search for cycling and hiking routes, view routes on a map, save routes to a wishlist and mark a route done.

MVP Users should be able to:

- Select start and finish locations for their route.
- Save routes to a wishlist.
- Mark completed routes as 'done'.

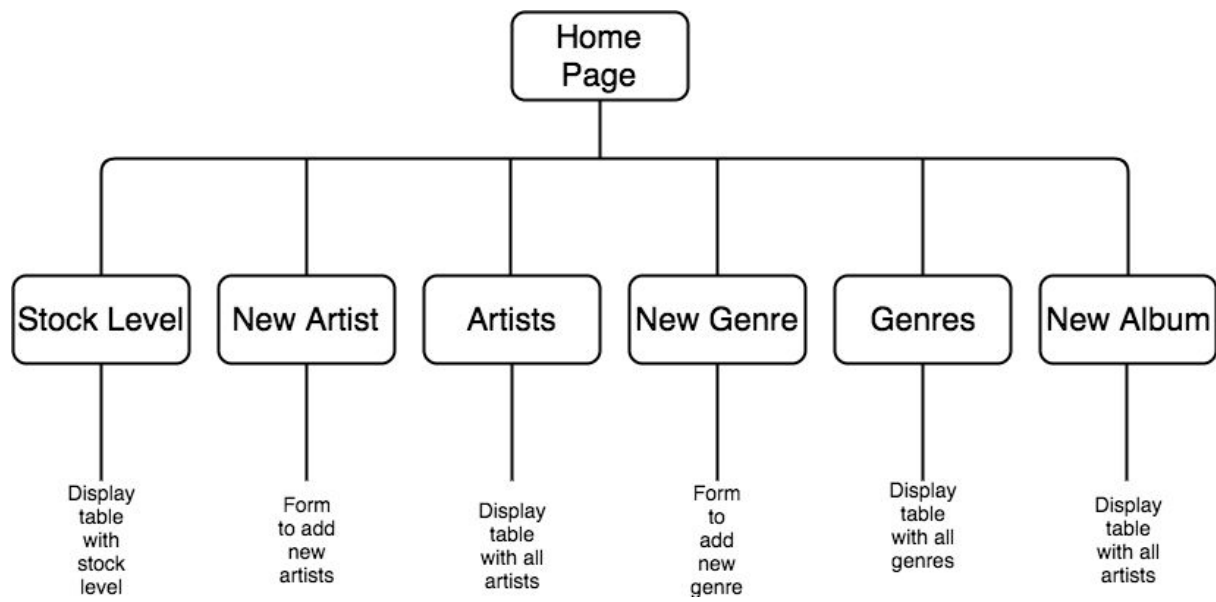
P. 3 Use of Trello



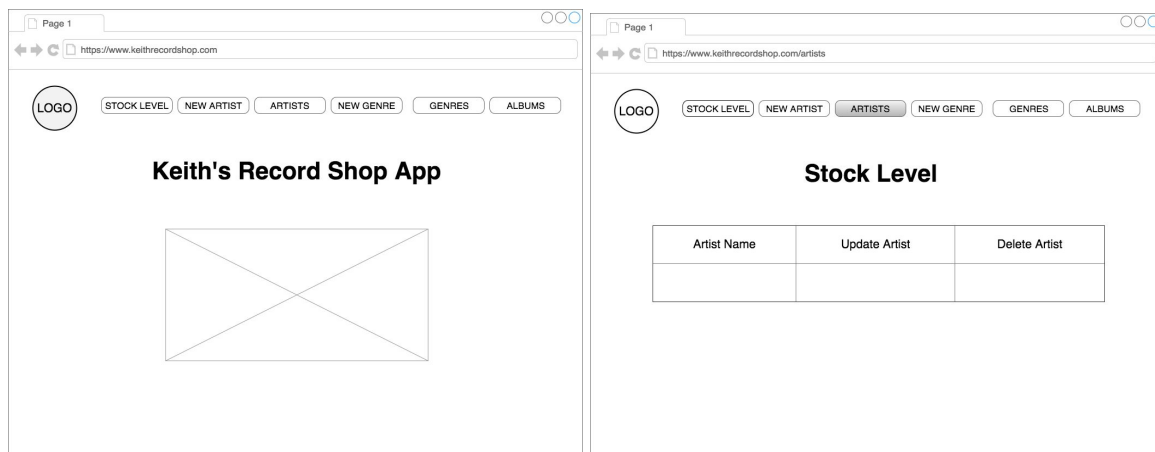
P. 4 Acceptance Criteria

Acceptance Criteria	Expected Result/Output	Pass/Fail
A user is able to save favourite routes to a wishlist	User click save button to save route to a wishlist	pass
A user is able to mark route as 'done'.	User check a checkbox to mark route as 'done'.	pass
A user is able to choose a start and end point to a route	User type into search field the start and end point.	pass
A user is able to share routes with friends	User click on share button and choose who to share route with from contact list	pass

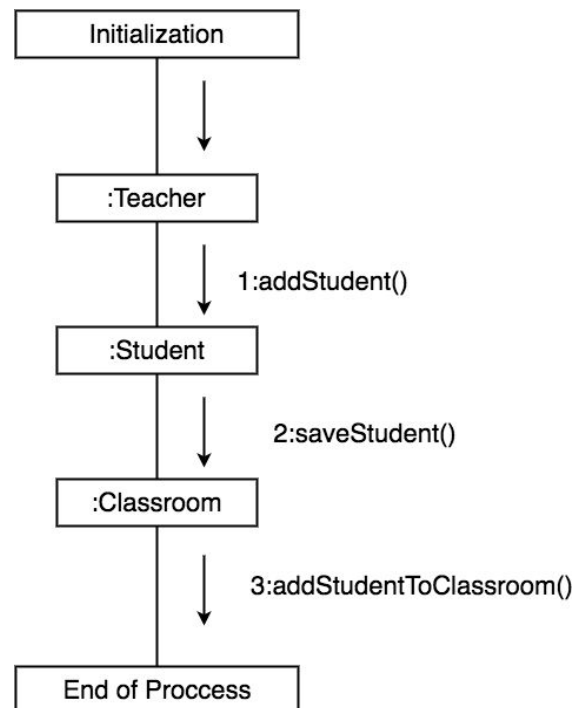
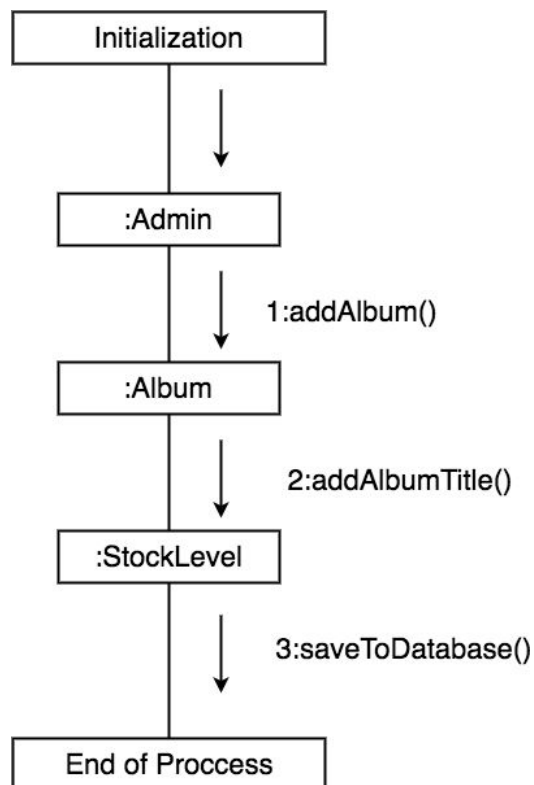
P. 5 User Sitemap



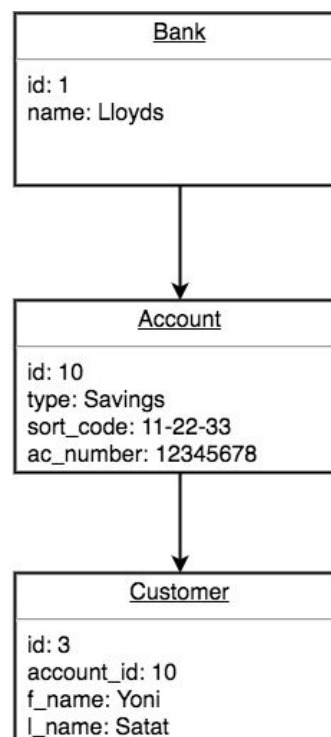
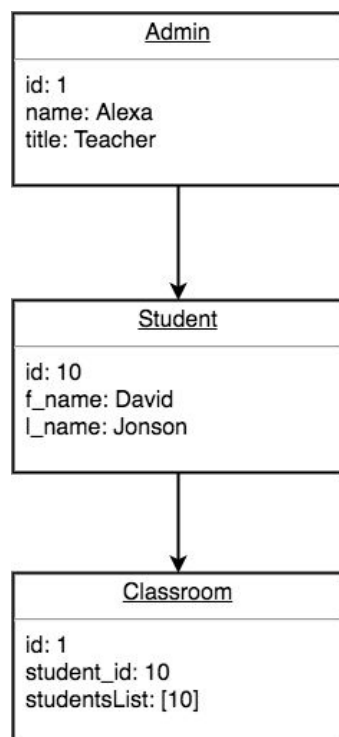
P. 6 Wireframes Design



P. 7 System Interactions Diagrams



P. 8 Two Object Diagrams



P. 9 Choice of two algorithms

This algorithm is to find a specific album in the database based on it's id.

I choose this algorithm because it is from my first project

```
def self.find(id)
  sql = "SELECT * FROM albums WHERE id = $1"
  values = [id]
  album = SqlRunner.run(sql, values)
  results = Album.new(album.first)
  return results
end
```

This algorithm is to find all albums saved into the database

I choose this algorithm because it is from my first project

```
def self.all()
  sql = "SELECT * FROM albums"
  values = []
  results = SqlRunner.run(sql, values)
  return results.map {|album| Album.new(album)}
end
```

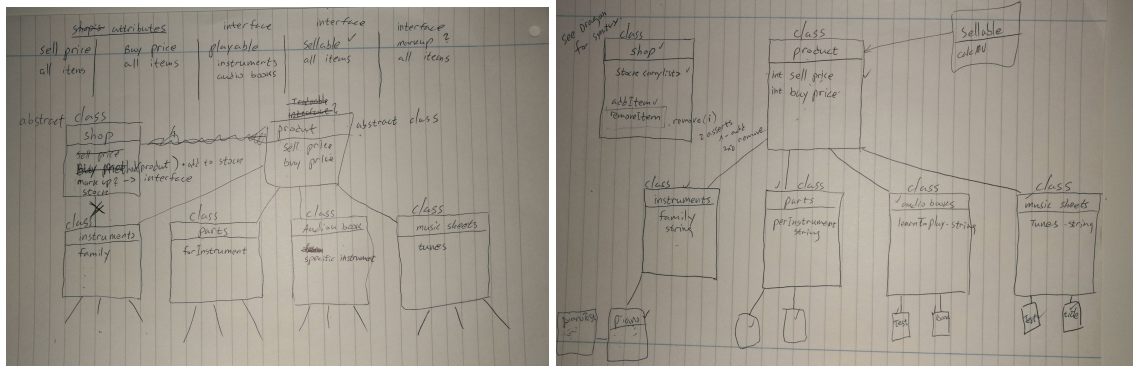
P. 10 Example for Pseudocode

```
# save new album to database
# create string with sql query
# create an array for the album's values
# pass all data to database via SqlRunner
```

P. 11 Github link to one of my projects

https://github.com/Yoni-Satat/ruby_project

P. 12 Screenshot of planning and different stages






P. 13 User input

Album Cover:	<input type="text" value="https://is3-ssl.mzstatic.com/image/thumb/Features"/>
Title:	<input type="text" value="Baloon"/>
Quantity:	<input type="text" value="20"/>
Select an Artist:	<input type="text" value="The Doors"/>
Select Genre:	<input type="text" value="Rock"/>
Sell Price:	<input type="text" value="10"/>
Buy Price:	<input type="text" value="5"/>
<input type="button" value="Add Album"/>	

P. 14 Interaction with data persistence

```
def save()
  sql = "INSERT INTO albums(title, quantity, artist_id, genre_id, sell_price, buy_price, image)
        VALUES($1, $2, $3, $4, $5, $6, $7)
        RETURNING id"
  values = [@title, @quantity, @artist_id, @genre_id, @sell_price, @buy_price, @image]
  results = SqlRunner.run(sql, values)
  @id = results.first()['id'].to_i
end
```

P. 15 User output result

	Baloon	High	The Doors	Rock	£10.0	£5.0	100.0		
---	--------	------	-----------	------	-------	------	-------	---	---

P. 16 Bug tracking report showing the errors diagnosed and corrected

User must be able to add new artist	Failed	Saving an artist using id	Passed
User must be able to delete artist	Failed	Delete artist by id	Passed
User must be able to view stock level	Failed	Finding all albums by their id	Passed
User must be able to update artist name	Failed	Update method added, finding artist by id and update in database	Passed

P. 17 Testing your program

```
calculator
  ✓ it has a sample test
  1) should start on zero
  2) should add a number
  3) should subtract a number
  ✓ should multiply
  ✓ should divide
  ✓ should know which number was clicked
  ✓ should know which operator was clicked
  ✓ should clear the screen from previews results

6 passing (11ms)
3 failing

1) calculator should start on zero:
ReferenceError: calculato is not defined
    at Context.<anonymous> (tests/unit/calculator_spec.js:15:24)

2) calculator should add a number:
AssertionError [ERR_ASSERTION]: undefined === 10
    at Context.<anonymous> (tests/unit/calculator_spec.js:20:12)

3) calculator should subtract a number:
TypeError: calculator.subtrct is not a function
    at Context.<anonymous> (tests/unit/calculator_spec.js:26:16)
```

```
14     it('should start on zero', function() {
15         assert.strictEqual(calculator.runningTotal, 0);
16     });
17
18     it('should add a number', function() {
19         calculator.add(10);
20         assert.strictEqual(calculator.runningTotal, 10);
21     });
22
23     it('should subtract a number', function() {
24         calculator.add(20);
25         calculator.operatorClick('+');
26         calculator.subtract(5);
27         assert.strictEqual(calculator.runningTotal, 15);
28     });
```

```
calculator
  ✓ it has a sample test
  ✓ should start on zero
  ✓ should add a number
  ✓ should subtract a number
  ✓ should multiply
  ✓ should divide
  ✓ should know which number was clicked
  ✓ should know which operator was clicked
  ✓ should clear the screen from previews results
```

9 passing (9ms)