Sprint 2 // Testing Document

User Story 1: As a user, I would like to be able to switch between dark mode and a light mode so that I can have an easier time viewing the application

- **Test:** Local storage updates correctly
    - Nature: Manual Testing
    - Expected: Local storage updates with the intended value of dark mode (true or false) based on what toggle was switched
- **Test:** Pages update with correct color scheme on dark mode toggle
    - Nature: Manual Testing
    - Expected: All pages are in dark mode when dark mode toggle is active and in light mode when light mode toggle is active

User Story 3: As a user, I would like to be able to save the stocks I'd like to watch so that I can quickly access the stocks that I have interest in.

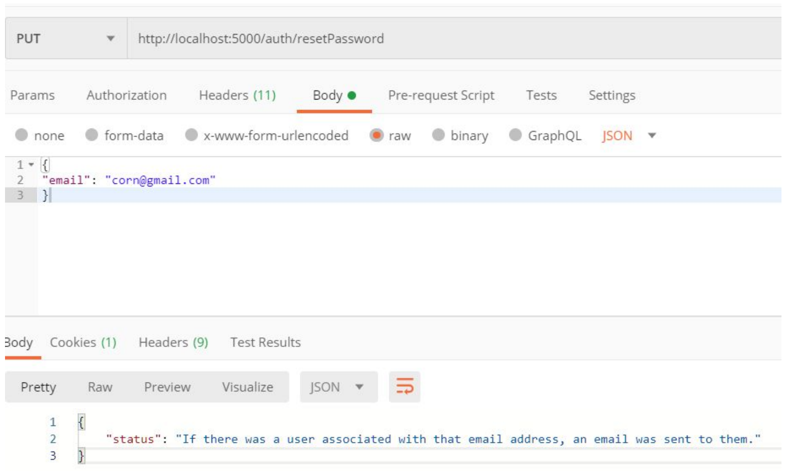
- **Test:** Stock info page renders correctly including no followed stocks when not logged in
    - Nature: Automatic Testing
    - Expected: The page renders correctly with no errors when logged in and not
- **Test:** Initial values for followed stocks are empty prior to load
    - Nature: Manual Testing
    - Expected: Before the user stocks are requested, nothing should be stored in the local state storage of the page

User Story 4: As a user, I would like to be able to create a monthly budget

- **Test:** The component renders correctly
    - Nature: Automatic Testing
    - Expected: No Errors are thrown when the test is run. Component is rendered correctly
- **Test:** The state of the component is set correctly when component is loaded
    - Nature: Automatic Testing
    - Expected: All initial values match the expected values.

User Story 5: As a user, I would like to be able to reset my password if I forget it

- **Test:** Reset password (valid email entered)
    - Nature: Manual Testing
    - Expected: 200 status with message saying if email was linked to an account, an email with a new password was sent, in JSON

- 
  PUT http://localhost:5000/auth/resetPassword

  Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings

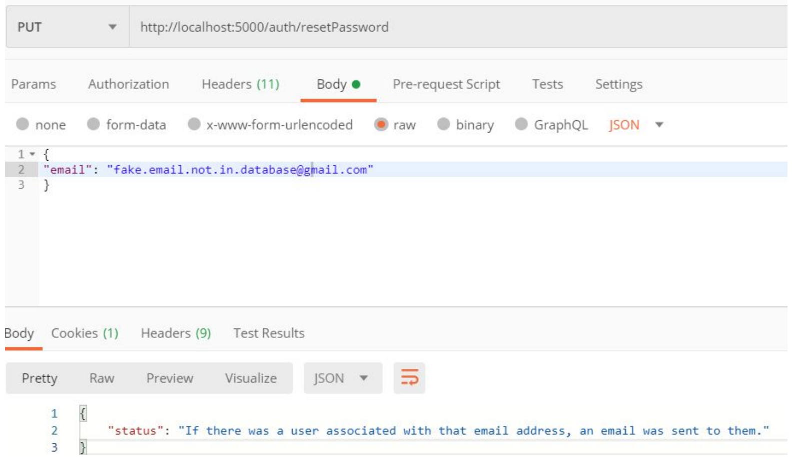
  none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

  ```
  {
  "email": "corn@gmail.com"
  }
  ```

  Body | Cookies (1) | Headers (9) | Test Results

  Pretty | Raw | Preview | Visualize | JSON

  ```
  {
      "status": "If there was a user associated with that email address, an email was sent to them."
  }
  ```

- **Test:** Reset password (invalid email entered)
  - Nature: Manual Testing
  - Expected: 200 status with message saying if email was linked to an account, an email with a new password was sent, in JSON (this is done so malicious people cannot tell if a certain email is linked to any account)
  - 
    PUT http://localhost:5000/auth/resetPassword

    Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings

    none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

    ```
    {
    "email": "fake.email.not.in.database@gmail.com"
    }
    ```

    Body | Cookies (1) | Headers (9) | Test Results

    Pretty | Raw | Preview | Visualize | JSON

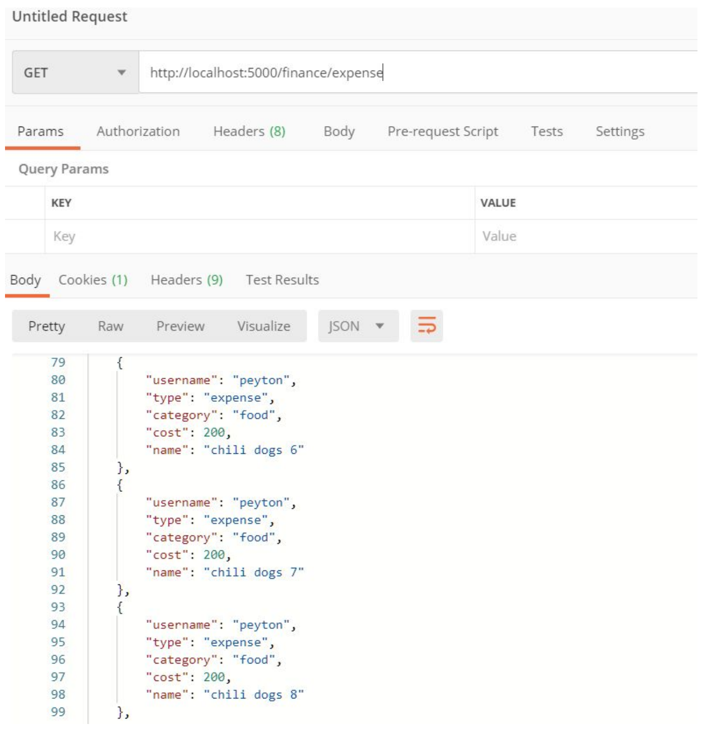
    ```
    {
        "status": "If there was a user associated with that email address, an email was sent to them."
    }
    ```

User Story 6: As a user, I would like to be able to edit my personal information

- **Test:** Update password (logged in)
  - Nature: Manual Testing
  - Expected: 200 status with message saying password was updated, in JSON
- **Test:** Update password (not logged in)
  - Nature: Manual Testing
  - Expected: 400 status with message saying not logged in, in JSON
- **Test:** Update email (logged in)
  - Nature: Manual Testing
  - Expected: 200 status with message saying email was updated, in JSON
- **Test:** Update email (not logged in)
  - Nature: Manual Testing
  - Expected: 400 status with message saying not logged in, in JSON

- **Test:** Update name (logged in)
    - Nature: Manual Testing
    - Expected: 200 status with message saying name was updated, in JSON
- **Test:** Update name (not logged in)
    - Nature: Manual Testing
    - Expected: 400 status with message saying not logged in, in JSON

User Story 7: As a user, I would like to be able to have a dashboard of all of my personal finances.

- **Test:** The component renders correctly
    - Nature: Automatic Testing
    - Expected: No Errors are thrown when the test is run. Component is rendered correctly
- **Test:** The state of the component is set correctly when component is loaded
    - Nature: Automatic Testing
    - Expected: All initial state values match expected values
- **Test:** Get expenses (logged in)
    - Nature: Manual Testing
    - Expected: 200 status with list of expenses associated with account, in JSON
    - 

Untitled Request

GET http://localhost:5000/finance/expense

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

Query Params

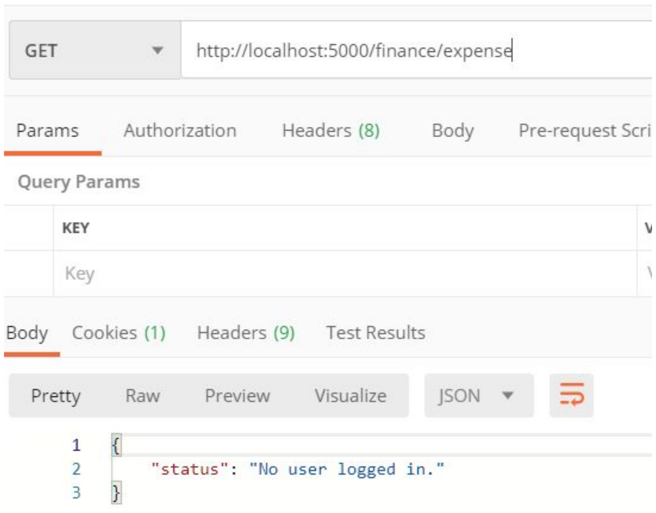
KEY	VALUE
Key	Value

Body | Cookies (1) | Headers (9) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
    {
        "username": "peyton",
        "type": "expense",
        "category": "food",
        "cost": 200,
        "name": "chili dogs 6"
    },
    {
        "username": "peyton",
        "type": "expense",
        "category": "food",
        "cost": 200,
        "name": "chili dogs 7"
    },
    {
        "username": "peyton",
        "type": "expense",
        "category": "food",
        "cost": 200,
        "name": "chili dogs 8"
    },
```

- **Test:** Get expenses (not logged in)
    - Nature: Manual Testing
    - Expected: 400 status with message saying not logged in, in JSON



GET http://localhost:5000/finance/expense

Params Authorization Headers (8) Body Pre-request Scri

Query Params

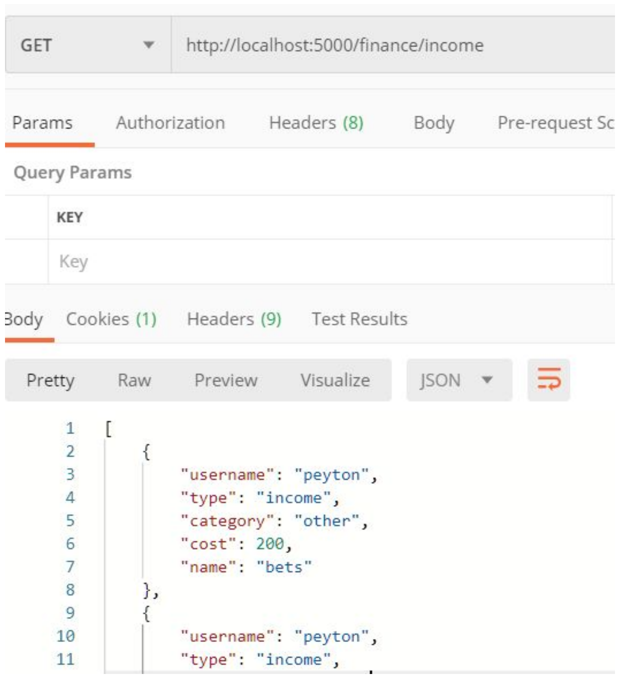
KEY
Key

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "status": "No user logged in."
}
```

-

- **Test:** Get income (logged in)
  - Nature: Manual Testing
  - Expected: 200 status with list of incomes associated with account, in JSON
  - 

GET http://localhost:5000/finance/income

Params Authorization Headers (8) Body Pre-request Sc

Query Params

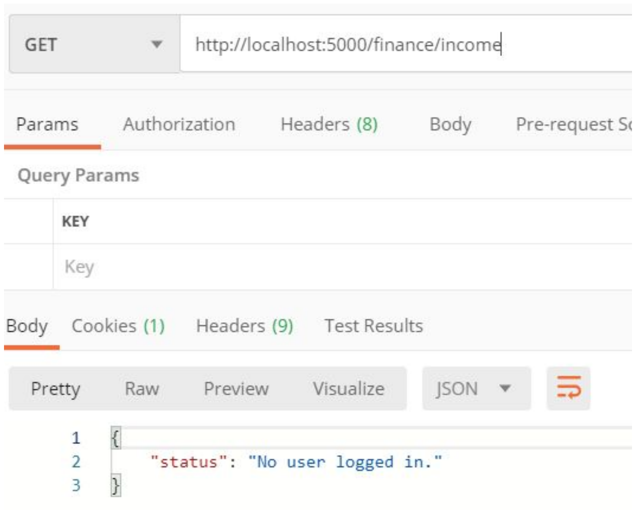
KEY
Key

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
[
    {
        "username": "peyton",
        "type": "income",
        "category": "other",
        "cost": 200,
        "name": "bets"
    },
    {
        "username": "peyton",
        "type": "income",
```

- **Test:** Get income (not logged in)
  - Nature: Manual Testing
  - Expected: 400 status with message saying not logged in, in JSON



GET http://localhost:5000/finance/income

Params Authorization Headers (8) Body Pre-request Sc

Query Params

KEY

Key

Body Cookies (1) Headers (9) Test Results

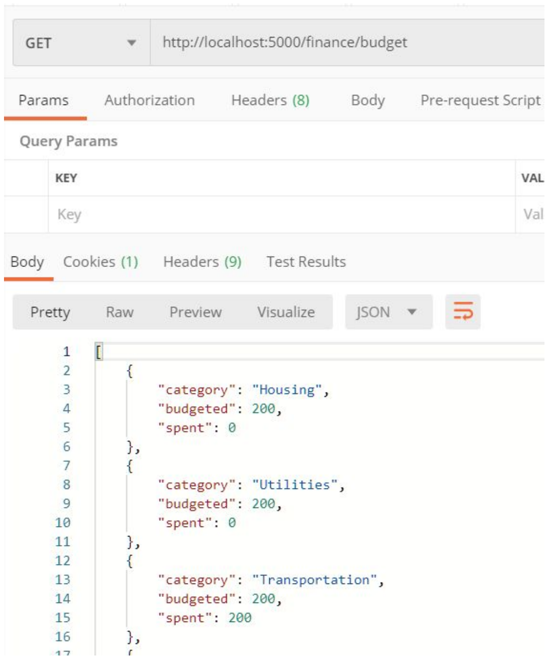
Pretty Raw Preview Visualize JSON

```
{
    "status": "No user logged in."
}
```

○

User Story 9: As a user, I would like to keep track of my actual spending as compared to my projected budget plan

- **Test:** Get Budget testing (logged in)
    - Nature: Manual Testing
    - Expected: 200 status with list of categories in each budget, with a budgeted and spent amount, in JSON



GET http://localhost:5000/finance/budget

Params Authorization Headers (8) Body Pre-request Script

Query Params

KEY	VAL
Key	Val

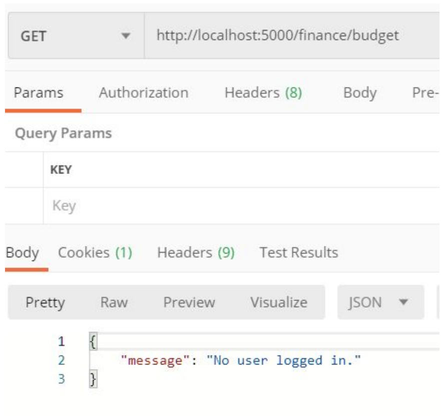
Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
[
    {
        "category": "Housing",
        "budgeted": 200,
        "spent": 0
    },
    {
        "category": "Utilities",
        "budgeted": 200,
        "spent": 0
    },
    {
        "category": "Transportation",
        "budgeted": 200,
        "spent": 200
    },
```

○

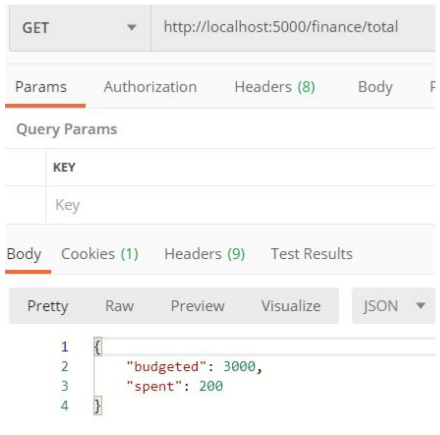
- **Test:** Get Budget testing (not logged in)
    - Nature: Manual Testing
    - Expected: 400 status with message saying not logged in, in JSON



http://localhost:5000/finance/budget

```
{
    "message": "No user logged in."
}
```

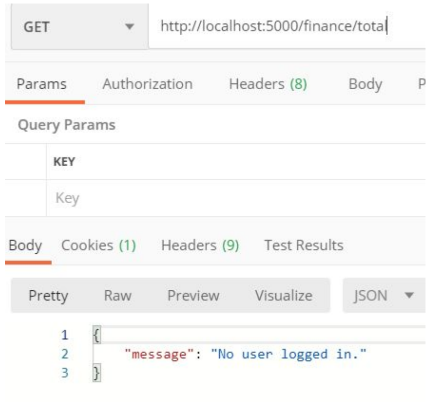
- **Test:** Get Total testing (logged in)
  - Nature: Manual Testing
  - Expected: 200 status with totals for budgeted and spent for the budget, in JSON



http://localhost:5000/finance/total

```
{
    "budgeted": 3000,
    "spent": 200
}
```

- **Test:** Get Total testing (not logged in)
  - Nature: Manual Testing
  - Expected: 400 status with message saying not logged in, in JSON



http://localhost:5000/finance/total

```
{
    "message": "No user logged in."
}
```

User Story 11: As a user, I would like to be able to view stock data in variable timeframes

- **Test:** Stock Intraday Expected Path
  - Nature: Automatic Testing

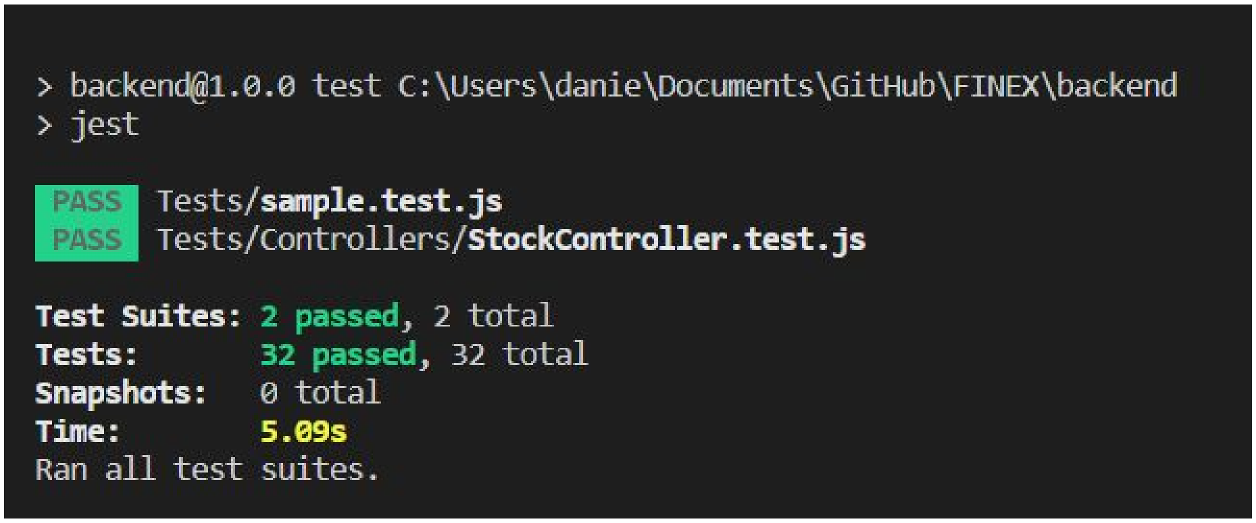
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** Stock Intraday Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** Stock Intraday stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message
- **Test:** Stock Daily Expected Path
    - Nature: Automatic Testing
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** Stock Daily Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** Stock Daily stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message

User Story 15: As a user, I would like to be able to view deep analytics on a stock so that I can better make buying and selling decisions

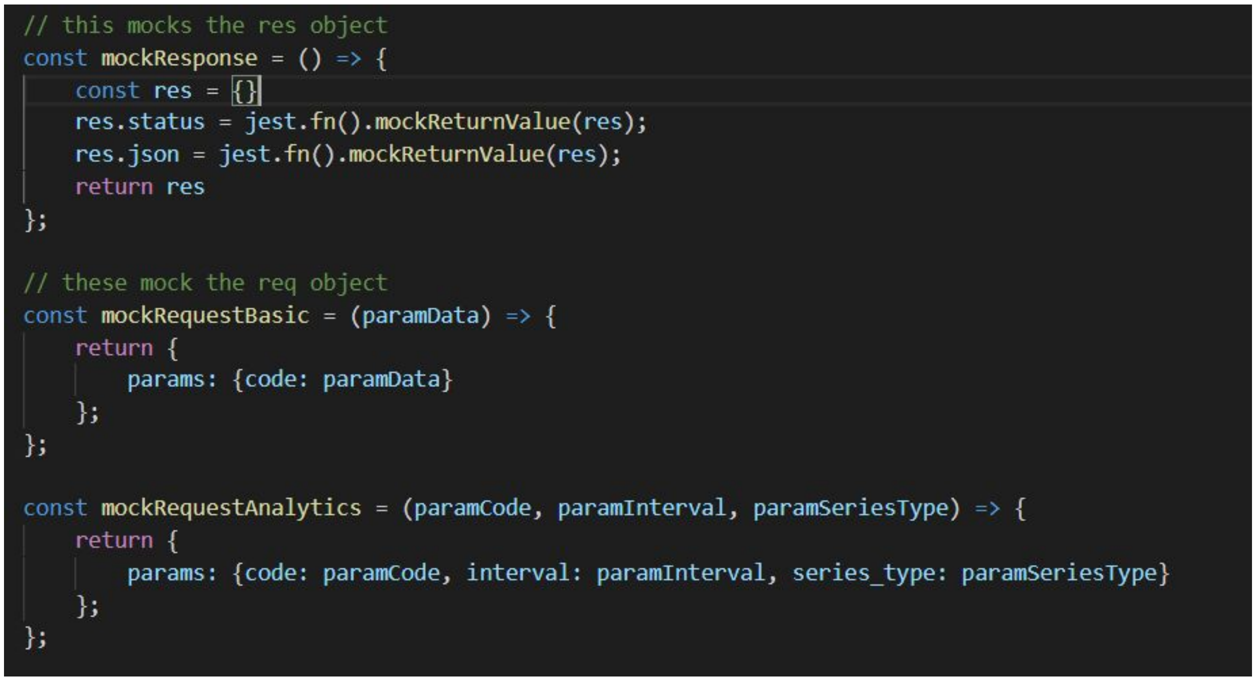
- **Test:** SMA Expected Path
    - Nature: Automatic Testing
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** SMA Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** SMA stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message
- **Test:** SMA stock interval is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid interval
    - Expected: 400 status with JSON containing “invalid interval” message
- **Test:** SMA series_type is invalid
    - Nature: Automatic Testing

    - Input: Request to stock controller for intraday data with invalid series_type
    - Expected: 400 status with JSON containing “invalid series_type” message
- **Test:** EMA Expected Path
    - Nature: Automatic Testing
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** EMA Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** EMA stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message
- **Test:** EMA stock interval is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid interval
    - Expected: 400 status with JSON containing “invalid interval” message
- **Test:** EMA series_type is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid series_type
    - Expected: 400 status with JSON containing “invalid series_type” message
- **Test:** RSI Expected Path
    - Nature: Automatic Testing
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** RSI Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** RSI stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message
- **Test:** RSI stock interval is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid interval
    - Expected: 400 status with JSON containing “invalid interval” message
- **Test:** RSI series_type is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid series_type
    - Expected: 400 status with JSON containing “invalid series_type” message
- **Test:** BBANDS Expected Path

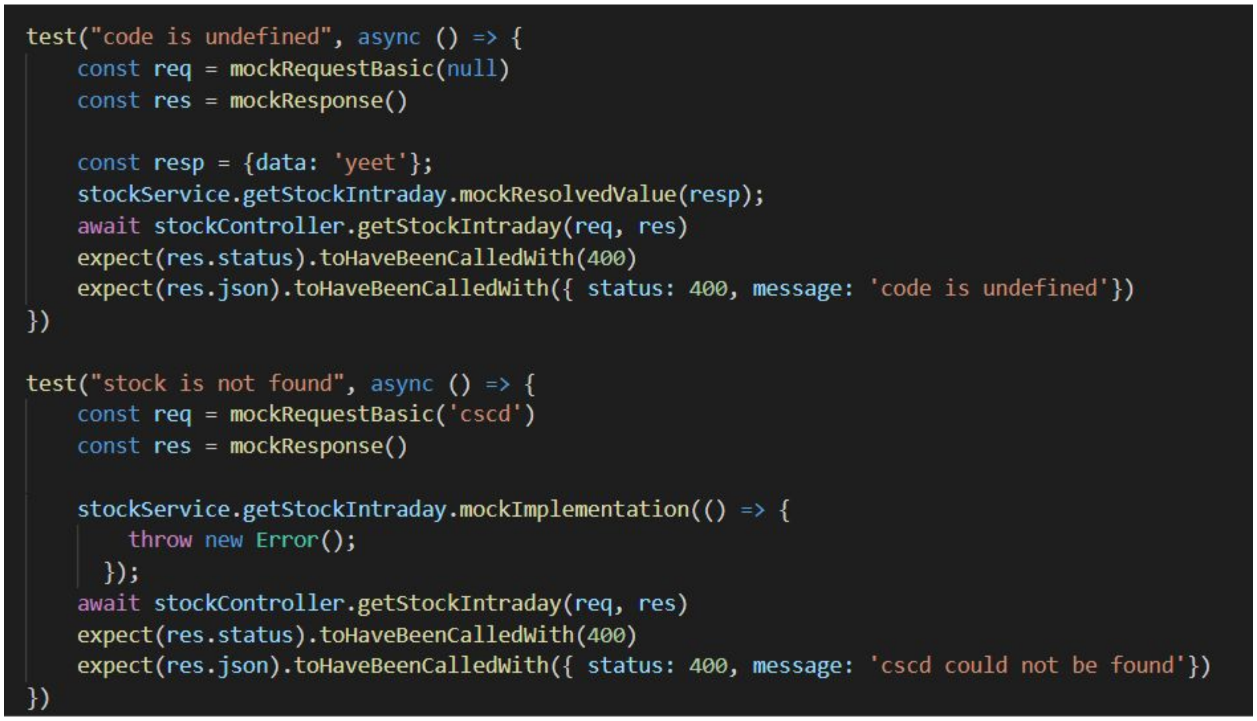
    - Nature: Automatic Testing
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** BBANDS Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** BBANDS stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message
- **Test:** BBANDS stock interval is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid interval
    - Expected: 400 status with JSON containing “invalid interval” message
- **Test:** BBANDS series_type is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid series_type
    - Expected: 400 status with JSON containing “invalid series_type” message
- **Test:** MACD Expected Path
    - Nature: Automatic Testing
    - Input: Valid request to stock controller for intraday data
    - Expected: 200 status with valid JSON returned from the service
- **Test:** MACD Code is undefined
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with null stock code
    - Expected: 400 status with json “stock undefined” error message
- **Test:** MACD stock is not found
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with fake stock
    - Expected: 400 status with JSON containing stock not found message
- **Test:** MACD stock interval is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid interval
    - Expected: 400 status with JSON containing “invalid interval” message
- **Test:** MACD series_type is invalid
    - Nature: Automatic Testing
    - Input: Request to stock controller for intraday data with invalid series_type
    - Expected: 400 status with JSON containing “invalid series_type” message



```
> backend@1.0.0 test C:\Users\danie\Documents\GitHub\FINEX\backend
> jest

 PASS  Tests/sample.test.js
 PASS  Tests/Controllers/StockController.test.js

Test Suites: 2 passed, 2 total
Tests:       32 passed, 32 total
Snapshots:   0 total
Time:        5.09s
Ran all test suites.
```



```
// this mocks the res object
const mockResponse = () => {
    const res = {}
    res.status = jest.fn().mockReturnValue(res);
    res.json = jest.fn().mockReturnValue(res);
    return res
};

// these mock the req object
const mockRequestBasic = (paramData) => {
    return {
        params: {code: paramData}
    };
};

const mockRequestAnalytics = (paramCode, paramInterval, paramSeriesType) => {
    return {
        params: {code: paramCode, interval: paramInterval, series_type: paramSeriesType}
    };
};
```



```javascript
test("code is undefined", async () => {
    const req = mockRequestBasic(null)
    const res = mockResponse()

    const resp = {data: 'yeet'};
    stockService.getStockIntraday.mockResolvedValue(resp);
    await stockController.getStockIntraday(req, res)
    expect(res.status).toHaveBeenCalledWith(400)
    expect(res.json).toHaveBeenCalledWith({ status: 400, message: 'code is undefined'})
})

test("stock is not found", async () => {
    const req = mockRequestBasic('cscd')
    const res = mockResponse()

    stockService.getStockIntraday.mockImplementation(() => {
        throw new Error();
      });
    await stockController.getStockIntraday(req, res)
    expect(res.status).toHaveBeenCalledWith(400)
    expect(res.json).toHaveBeenCalledWith({ status: 400, message: 'cscd could not be found'})
})
```