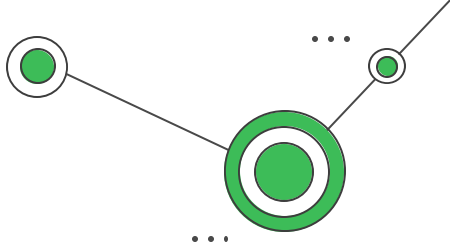
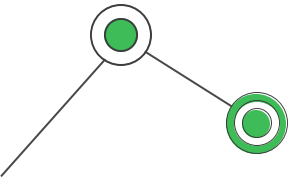
# Let’s apply what we have learned!

# ATDD Exercises



- In **assigned pairs**, do **ATDD** to implement the following behaviour that your PO has request.

- Commit your code once you are finished.



# PO Requests

01 When my account is credited, the credited amount should appear in my account. ✓

02 As a user, I want to withdraw money from my account. If I have no money in my account, I should not be allowed to withdraw any money.

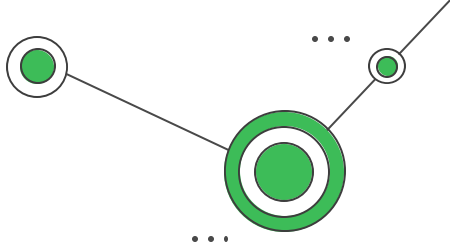
- ***Want you to use a Scenario Outline, where a few different amounts can be tested against the same scenario***

03 As a user, I want to change the name of my account. The account name must not have any numbers in it.

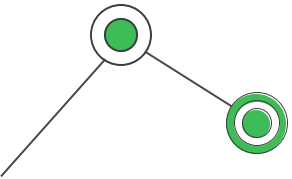
04 To deter me from going on a shopping spree, as a user, I can set the maximum number of times I can withdraw money from my account within the duration of a day.

05 As a user, if a maximum number of daily withdraws is set, then I am not allowed to exceed this number of withdraws with a single day.

06 As a user, if I try exceed the number of allowed daily withdraws, I get charged a fine of $50.



# How do we do ATDD for more complex scenarios?



# AC5 – Setting the Granularity of a Sales Report

## U41 Sales report (depends on U31)

So that I can manage my business effectively, as a logged-in business admin, I need to be able to see how much has been sold in a period of time.

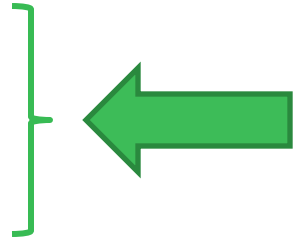
AC1: From some suitable location (e.g. my home page or business profile) I can access the report feature.

AC2: I can select a period to be reported on. This might be a single year, month, week or day.

AC3: I can also specify a custom period by selecting when it starts and ends.

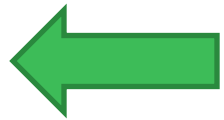
AC4: I can select the granularity of the report. By default, I will just see the total number and total value of all purchases made during the period, together with the details of the period, and any other relevant detail (e.g. the business name)..

AC5: I can also select finer granularity (e.g. monthly). In this case the report would have a line for each month, including the month name/number and the correspondent total number and total value for that month.

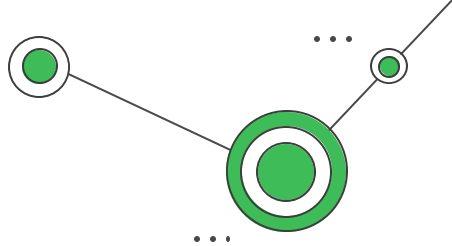


# Scenario – Setting the Granularity of a Sales Report

```
Scenario: AC5 - I can also select finer granularity.
  Given I am logged in as an administrator of an existing business.
  When I select a granularity of "Daily" for the sales report.
  Then A sales report is returned with the "Daily" granularity.
```



# Scenario – Setting the Granularity of a Sales Report



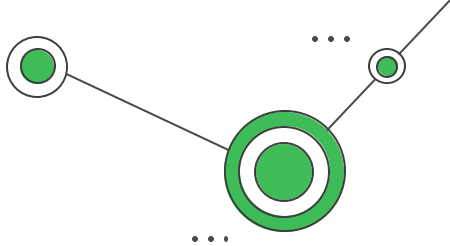
```
public class SalesReportStepDefs extends CucumberSpringConfiguration {

    @Autowired
    private MockMvc mvc;

    @Autowired
    @MockBean
    private UserRepository userRepository;

    @Autowired
    @MockBean
    private BusinessRepository businessRepository;

    @Autowired
    @MockBean
    private ProductRepository productRepository;

    @Autowired
    @MockBean
    private InventoryItemRepository inventoryItemRepository;

    @Autowired
    @MockBean
    private ListingRepository listingRepository;
```

```
    @Autowired
    @MockBean
    private SoldListingRepository soldListingRepository;

    @Autowired
    @MockBean
    private ListingNotificationRepository listingNotificationRepository;

    @Autowired
    @MockBean
    private SoldListingNotificationRepository soldListingNotificationRepository;

    @Autowired
    @MockBean
    private BookmarkedListingMessageRepository bookmarkedListingMessageRepository;

    private User user;

    private Business business;
```

# GIVEN

```java
@Given("I am logged in as an administrator of an existing business.")
public void iAmLoggedInAsAnAdministratorOfAnExistingBusiness() throws Exception {
    Address address = new Address(
            "3/24",
            "Ilam Road",
            "Christchurch",
            "Canterbury",
            "New Zealand",
            "90210",
            "Ilam"
    );

    user = new User(
            "John",
            "Doe",
            "S",
            "Generic",
            "Biography",
            "email@email.com",
            LocalDate.of(2000, 2, 2),
            "0271316",
            address,
            "Password123!",
            LocalDateTime.of(LocalDate.of(2021, 2, 2),
                    LocalTime.of(0, 0)),
            Role.USER);
    user.setId(1);
    user.setSessionUUID(User.generateSessionUUID());
```

# GIVEN Continued



```
business = new Business(
        user.getId(),
        "name",
        "some text",
        address,
        BusinessType.ACCOMMODATION_AND_FOOD_SERVICES,
        LocalDateTime.of(LocalDate.of(2021, 2, 2), LocalTime.of(0, 0)),
        user,
        "$",
        "NZD"
);
business.setId(1);
```

```
product = new Product(
        "WATT-420-BEANS",
        business,
        "Beans",
        "Description",
        "Manufacturer",
        20.99,
        "9400547002634"
);

inventoryItem = new InventoryItem(
        product,
        "WATT-420-BEANS",
        100,
        20.99,
        2099.00,
        LocalDate.of(2021, 1, 1),
        LocalDate.of(2023, 1, 1),
        LocalDate.of(2023, 1, 1),
        LocalDate.of(2024, 1, 1)
);
```

```
    listing = new Listing(
            inventoryItem,
            10,
            55.55,
            "info",
            LocalDateTime.now(),
            null
    );

    soldListing = new SoldListing(
            business,
            user,
            LocalDateTime.of(2021, Month.JUNE, 11, 0, 0),
            listing.getInventoryItem().getProduct().getProductId(),
            listing.getQuantity(),
            listing.getPrice(),
            0
    );
    soldListing.setId(1);
    soldListing.setSaleDate(LocalDateTime.of(2021, Month.JULY, 20, 0, 0));

    user.setBusinessesAdministeredObjects(List.of(business));

    given(userRepository.findBySessionUUID(user.getSessionUUID())).willReturn(Optional.ofNullable(user));
    given(businessRepository.findBusinessById(business.getId())).willReturn(Optional.ofNullable(business));
}
```

# THEN & WHEN

```
@When("I select a granularity of {string} for the sales report.")
public void iSelectAGranularityOfForTheSalesReport(String granularityInput) { granularity = granularityInput; }

@Then("A sales report is returned with the {string} granularity.")
public void aSalesReportIsReturnedWithTheGranularity(String granularityInput) throws Exception {
    response = mvc.perform(get(String.format("/businesses/%d/salesReport", business.getId()))
                    .cookie(new Cookie("JSESSIONID", user.getSessionUUID()))
                    .param("fromDate", "2021-04-20T00:00")
                    .param("toDate", "2021-09-30T00:00")
                    .param("granularity", granularityInput))
            .andReturn().getResponse();
    List<SalesReportPayload> responseList = mapper.readValue(
            response.getContentAsString(), new TypeReference<>(){}
    );
    List<String> granularityNames = List.of(
            "April 2021", "May 2021", "June 2021", "July 2021", "August 2021", "September 2021"
    );

    assertThat(response.getStatus()).isEqualTo(HttpStatus.OK.value());
    for (int i = 0; i < responseList.size(); i++) {
        assertThat(responseList.get(i).getGranularityName()).isEqualTo(granularityNames.get(i));
    }
}
```

# Writing Code to Pass Tests – Part 1

```java
/**
 * Retrieve sales report for a business, by from/to dates and granularity (e.g. Yearly).
 *
 * @param sessionToken Session token used to authenticate user (is user logged in?).
 * @param businessId ID of the business to retrieve the sales report from.
 * @param fromDate The date the sales report should be from.
 * @param toDate The date the sales report should be to.
 * @param granularity The granularity of the sales report (e.g. Yearly).
 * @return List of sales report payloads containing granularity name, total sales and total revenue.
 */
@GetMapping("/businesses/{businessId}/salesReport")
public ResponseEntity<List<SalesReportPayload>> retrieveSalesReport(
        @CookieValue(value = "JSESSIONID", required = false) String sessionToken,
        @PathVariable Integer businessId,
        @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime fromDate,
        @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime toDate,
        @RequestParam(defaultValue = "Total") String granularity
) {
    logger.debug(
            "Business sales report request received with business ID {}, from date {}, " +
                    "to date {}, granularity {}",
            businessId, fromDate, toDate, granularity);

    // 401 if not verified
    User user = Authorization.getUserVerifySession(sessionToken, userRepository);

    // 406 if business does not exist
    Authorization.verifyBusinessExists(businessId, businessRepository);

    // 403 if not a business admin nor a GAA
    Authorization.verifyBusinessAdmin(user, businessId);

    LocalDateTime startOf2021 = LocalDateTime.of(2021, Month.JANUARY, 1, 0, 0);
    if (fromDate.isBefore(startOf2021)) {
        fromDate = startOf2021;
    }
    if (toDate.isAfter(LocalDateTime.now())) {
        toDate = LocalDateTime.now();
    }
```

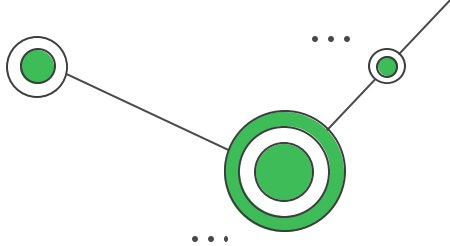
# Writing Code to Pass Tests – Part 2

```
// 400 if granularity does not exist
ArrayList<SalesReportPayload> salesReportPayloads = new ArrayList<>();
LocalDateTime currentDate = fromDate;
switch (granularity) {
    case "Total":
        salesReportPayloads.add(generateIndividualSalesReport(businessId, fromDate, toDate, null));
        break;
    case "Yearly":
        while (currentDate.getYear() != toDate.getYear()) {
            salesReportPayloads.add(generateIndividualSalesReport(
                    businessId, currentDate, currentDate.with(lastDayOfYear()),
                    String.valueOf(currentDate.getYear())
            ));
            currentDate = currentDate.plusYears(1).with(firstDayOfYear());
        }
        salesReportPayloads.add(generateIndividualSalesReport(
                businessId, currentDate, toDate, String.valueOf(currentDate.getYear())
        ));
        break;
    case "Monthly":
        while (currentDate.getYear() != toDate.getYear() || currentDate.getMonth() != toDate.getMonth()) {
            salesReportPayloads.add(generateIndividualSalesReport(
                    businessId, currentDate, currentDate.with(lastDayOfMonth()),
                    currentDate.getMonth().getDisplayName(TextStyle.FULL, Locale.ENGLISH) + " " +
                            currentDate.getYear()
            ));
            currentDate = currentDate.plusMonths(1).with(firstDayOfMonth());
        }
        salesReportPayloads.add(generateIndividualSalesReport(
                businessId, currentDate, toDate,
                currentDate.getMonth().getDisplayName(TextStyle.FULL, Locale.ENGLISH) + " " +
                        currentDate.getYear()
        ));
        break;
```

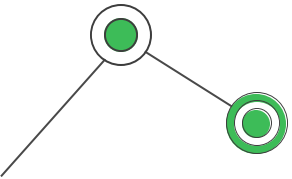
# Writing Code to Pass Tests – Part 3

```
            case "Weekly":
                while (
                        currentDate.getYear() != toDate.getYear() ||
                        currentDate.get(WeekFields.of(Locale.US).weekOfWeekBasedYear()) !=
                                toDate.get(WeekFields.of(Locale.US).weekOfWeekBasedYear())
                ) {
                    salesReportPayloads.add(generateIndividualSalesReport(
                            businessId, currentDate, currentDate.with(DayOfWeek.SUNDAY),
                            "Week " + currentDate.get(WeekFields.of(Locale.US).weekOfWeekBasedYear()) + ", " +
                                    currentDate.getYear()
                    ));
                    currentDate = currentDate.plusWeeks(1).with(DayOfWeek.MONDAY);
                }
                salesReportPayloads.add(generateIndividualSalesReport(
                        businessId, currentDate, toDate,
                        "Week " + currentDate.get(WeekFields.of(Locale.US).weekOfWeekBasedYear()) + ", " +
                                currentDate.getYear()
                ));
                break;
            case "Daily":
                while (
                        currentDate.getYear() != toDate.getYear() ||
                        currentDate.getMonth() != toDate.getMonth() ||
                        currentDate.getDayOfMonth() != toDate.getDayOfMonth()
                ) {
                    salesReportPayloads.add(generateIndividualSalesReport(
                            businessId, currentDate, currentDate.with(LocalTime.MAX),
                            currentDate.getDayOfMonth() + " " +
                                    currentDate.getMonth().getDisplayName(TextStyle.FULL, Locale.ENGLISH) + " " +
                                    currentDate.getYear()
                    ));
                    currentDate = currentDate.plusDays(1).with(LocalTime.MIN);
                }
                salesReportPayloads.add(generateIndividualSalesReport(
                        businessId, currentDate, toDate,
                        currentDate.getDayOfMonth() + " " +
                                currentDate.getMonth().getDisplayName(TextStyle.FULL, Locale.ENGLISH) + " " +
                                currentDate.getYear()
                ));
                break;
```

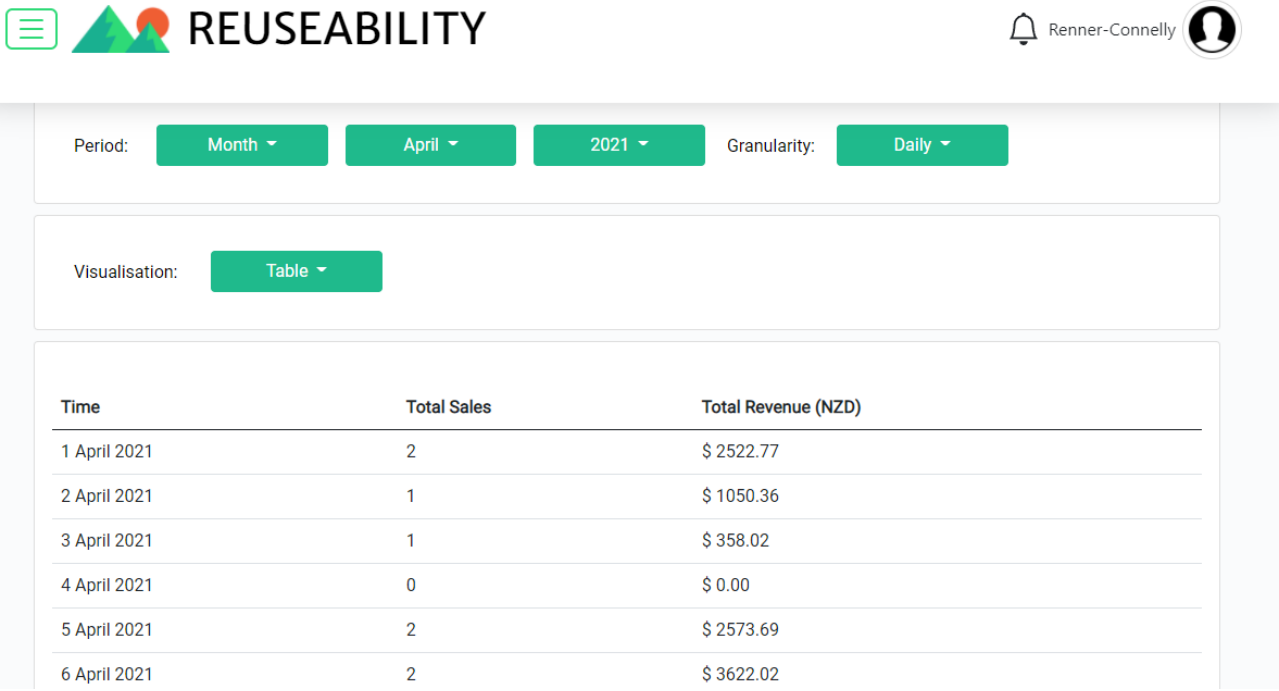
# Writing Code to Pass Tests – Part 4



```
            break;
        default:
            logger.error("400 [BAD REQUEST] - Granularity type {} does not exist", granularity);
            throw new ResponseStatusException(
                    HttpStatus.BAD_REQUEST, "There was some error with the data supplied."
            );
    }

    logger.info(
            "Sales Report Success - 200 [OK] - Sales Report retrieved for business ID {}, from date {}, " +
                    "to date {}, granularity {}",
            businessId, fromDate, toDate, granularity
    );
    return ResponseEntity.ok().body(salesReportPayloads);
}
```



# Functionality




| Time | Total Sales | Total Revenue (NZD) |
|---|---|---|
| 1 April 2021 | 2 | $ 2522.77 |
| 2 April 2021 | 1 | $ 1050.36 |
| 3 April 2021 | 1 | $ 358.02 |
| 4 April 2021 | 0 | $ 0.00 |
| 5 April 2021 | 2 | $ 2573.69 |
| 6 April 2021 | 2 | $ 3622.02 |

# Before Part 2 of ATTD Workshop (Tomorrow)

- Set up your own projects with Cucumber before coming to the second workshop.
- Use the instructions from SENG301 lab or, base it on the account exercise's set up

# Apply ATDD to Your Project

- Same pairs as the exercise about banking.

- Select an aspect of your project that needs to be implemented.
- In your team's wiki, note what area you will apply ATDD.
- Use ATDD to write 2 scenarios.
- Remember that some ACs may require multiple scenarios.
- Use tags as you add scenarios so that you can see how you can execute ATDD only for a story.

- In your team's wiki, document all scenarios implemented (how far you got -scenarios written, test implemented, feature implemented). Write what issues you faced if you didn't finish at least 2 scenarios by the end of this workshop.