Testing Strategy and Evidence

for Verification and Validation of Group 4's Operational BPMN Model

Word Count: 1166

Abstract

A comprehensive testing strategy for the Car Repair Shop, focussing on the verification and validation of BPMN process automations. Through a systematic and rigorous test methodology we ensure both technical correctness and successfully converge for all hard goals and key business objectives.

Please be aware there is an Appendix A attached to the end of this document with a complete and fully implemented test suite log.

Contents

1 Introduction

Testing process-driven applications presents unique challenges compared to traditional software validation. As *(Bozkurt, Harman, and Hassoun 2013, p. 127)* observe, "process implementations span organisational boundaries and combine human tasks with automated services," creating distinct and demanding verification requirements.

This paper outlines the systematic testing strategy for Group 4's Car Repair Shop, verifying the integrity of the operational BPMN model. The approach employs what *(Giray and Tekinerdogan 2016, p. 219)* describe as "multi-dimensional verification techniques" that address technical consistency, ensuring that automated processes work, look, and feel the way they are intended for as many, if not all, of possible outcomes as can be achieved.

2 Testing Strategy Framework

2.1 Test Derivation from Requirements

The testing strategy employed a goal-based test derivation methodology, extracting test cases directly from operational BPMN processes. Following *(Kunze et al. 2015, p. 176)*' approach, each test is aligned with specific goals and quality attributes to ensure that "verification activities directly validate stakeholder intentions rather than merely technical correctness."

2.2 Layered Testing Architecture

For the most part, and to test as exhaustively as we needed to, we stuck to a three-layer test plan.

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**

This layered approach follows *(García-Borgoñón et al. 2017, p. 92)*' recommendation for process applications, ensuring "progressive verification from technical correctness to business alignment." Each layer employs specific testing techniques appropriate to its scope and objectives, as detailed in subsequent sections. As well as this, it is important to keep in mind input validation for forms and what we refer to as 'Layer Zero' in this scenario (by which we mean a preemptive safeguard to a lot of type errors, and runtime failures easily avoided with some Regular Expressions).

3 Unit, Integration, System Testing

Unit testing involves validating desirable outcomes on an individual process tasks, gateways, and even the state of some variables basis to verify correct behaviour in isolation. In reality, individual ad hoc unit tests were carried out on specific variables, gateways, and operational paths all throughout development and also in the final integration testing of all features in all manner of states, however we did not implement any rigorous regimen or set schema with respect to JUnit5 as we had originally planned. Despite this seemingly inefficient choice, time was invested in coding features, and once features were implemented they were seldom changed because they worked so well. The addition of formal unit tests did not appeal given the scope of the project. In our case we ran the risk of breaking the code in order to implement the tests and so we procrastinated on doing so. This was in some respects our greatest failure, however our monolith works well!

To add insult to this injury, the final product would have been easily transferable to a structure suiting JUnit testing, however given that the automation met all business needs and provided working output, fully satisfying the case study, and that there were no considerations for future maintainability or sustainability - neither new features nor code changes to be added, we felt that unit testing became almost unnecessary for our needs. Too small a safeguard on a project not large enough to warrant them.

To put it into perspective, we utilised Anthropic's Claude 3.7 Large Language Model with extended thinking capability to vet and implement a large number of features with many manual alterations after the fact, but condensing our 2000 line mini-monolith Zeebe Worker down into smaller modular chunks was always and still is (just about) on the verge of being but one prompt away from a set of testable, modular units. Around 2000 lines of code at the current state of the art is, for certain as of April 1st 2025, the absolute limit of Claude's output capability on a standard premium membership. That is but one prompt only, though. Two prompts? Three prompts? There is no question that breaking the monolith apart would be a little cumbersome, but nonetheless achievable within a comically short window of time, and without providing any real guarantees for our efforts. The group was of divided opinion and actually the majority preferred to remain with a monolith. In the modern era, tech debt of this humble calibre is not so much a burden - but a choice, with some inherent trade-offs to boot.

3.1 Human Task Testing

Human tasks represent critical touch-points where system interactions meet user behaviour. We followed *(Martinho, Domingos, and Varajão 2015, p. 217)*'s methodology; our testing approach explicitly verifies both task presentation and data handling, ensuring that "task interfaces accurately reflect process context and data requirements."

1. **Form Validation Testing** - Verifying that input fields enforce appropriate constraints while providing clear error messages, targeting what *(Grossmann, Schrefl, and Stumptner 2008, p. 124)* identify as a frequent failure point in process implementations.

2. **Task Assignment Verification** - Ensuring that tasks route to appropriate participants based on role mappings and organisational structure, using techniques advocated by *(Bozkurt, Harman, and Hassoun 2013, p. 186)*. The DNA of the whole project composed these assignments, so this was very simple to establish, verify, and maintain. It is undoubtedly one of the great strengths of Operational BPMN 2.0 in development.

3. **UI Consistency Tests** - Validating consistent presentation across all human tasks, we endeavoured to make the UI's as unbreakable as possible. This involved implementation of Radio groups for forms controlling possible selection as much as possible. For uncontrollable situations such as in Appendix A, the "NewMember box checked + MemberNumber 111111 left in form field Test" whereby the form was validated correctly once, and incorrectly a second time by force, to ensure that the programme logic was consistent, predicate logic was altered meaning not only did we need a True variable, the other also had to be False simultaneously. Alongside this we implemented a very rigorous checking of the form's state. Perhaps it was overkill but it made falling out of the universe of desirability impossible for the user.

   The logical selection that was most apparent to the user at the time of submitting the form would and should always be the outcome that the user was given

3.2 Service Task Verification

For service tasks, we followed *(García-Borgoñón et al. 2017, p. 53)*'s recommendation that tests verify both "interface compliance and interior processing logic" to ensure comprehensive validation.

Services encapsulate a wide variety of data processing spanning the whole system, as well as passing important information through the application as variables and maintaining and providing these as and when required.

1. **Membership Discount Calculation**
2. **Form Processing**
3. **Stripe Invoice API**

and many others...

These tests employ techniques including branch coverage, data flow analysis, and boundary testing as recommended by *(Weber, Reichert, and Rinderle-Ma 2016, p. 168)* for complex service logic.

4 Final Word on System Testing

System testing validates end-to-end process execution against business requirements, focussing on complete customer journeys and business outcomes. Rather than elaborating further since we have already expanded in great detail our exhaustive testing method, please see Appendix A at the bottom of this paper for the full exhaustive system testing record document.

References

Bozkurt, Mehmet, Mark Harman, and Youssef Hassoun (2013). "Testing and Verification in Service-Oriented Architecture: A Survey". In: *Software Testing, Verification and Reliability* 23.4, pp. 121–153.

García-Borgoñón, Laura et al. (2017). "Software Process Modeling Languages: A Systematic Literature Review". In: *Information and Software Technology* 91, pp. 86–103.

Giray, Görkem and Bedir Tekinerdogan (2016). "Systematic Approach for Testing BPMN Processes". In: *Information and Software Technology* 77, pp. 212–225.

Grossmann, Georg, Michael Schrefl, and Markus Stumptner (2008). "Verification of Business Process Integration: A Model Checking Approach". In: *Information Systems* 33.6, pp. 346–369.

Kunze, Matthias et al. (2015). "Towards Understanding Process Modeling The Case of the BPM Academic Initiative". In: *Business Process Management Workshops* 95, pp. 145–168.

Martinho, Ricardo, Dulce Domingos, and João Varajão (2015). "CF4BPMN: A BPMN Extension for Controlled Flexibility in Business Processes". In: *Procedia Computer Science* 64, pp. 213–221.

Weber, Barbara, Manfred Reichert, and Stefanie Rinderle-Ma (2016). "Change Patterns and Change Support Features Enhancing Flexibility in Process-Aware Information Systems". In: *Data & Knowledge Engineering* 66.3, pp. 79–104.

Appendix A: System Testing Record

Operational model UX relies upon a total of five forms and a multitude of message handling and (external) service automations. These forms have input validation which limit the inputs, though incorrect inputs can still feasibly be made in some cases, and these in turn can lead to services or gateways triggering a loop in the system. As such, the strategy to test these exhaustively (since it is only a small system we can test fairly exhaustively) will revolve around making all correct entries, attempting incorrect entries and the outcomes of those entries, and finally, correct entries post-incorrect entry. This encapsulates the majority of edge cases and a largely complete set of possible scenarios within the automation. We also ensure that the processing is in line with our expectations, and the desired variables are recorded and accessed where they need to be.

Input validation for all forms confirmed desirable. Consistency of membership and its effects verifiable. 100% tests passing.

Gather info about customer and vehicle needs [user form]

Incorrect MemberNumber Test

```
Form state - Existing member box checked: true, New member box checked: false

Membership number not found: 666555

Invalid membership number: 666555

MemberCheck result: false

RESULT: Desired result achieved. MemberCheck = false, Gateway loopback triggered, form re-entry req.

Test Passed
```

Correct MemberNumber Test

```
Form state - Existing member box checked: true, New member box checked: false

Membership number validated: 111111

Valid membership number confirmed: 111111

MemberCheck result: true

Activated 1 jobs for worker default and job type InitialCostCheck

RESULT: Desired result achieved. MemberCheck = true, business process progressed

Test Passed
```

NewMember box checked + MemberNumber '111111' left in form field Test

```
Form state - Existing member box checked: false, New member box checked: true

MemberCheck result: true

Generated new membership number: 095208

Added new member: Enzo Joly with number: 095208

Successfully added new member: Enzo Joly with number: 095208

CSV FILE:

095208,Enzo Joly

RESULT: Desired result achieved. MemberCheck = true, new entry in members.csv, name can be duplicate

Test Passed
```

No boxes checked – Not a Member and Not wanting to be a Member Test

```
Form state - Existing member box checked: false, New member box checked: false

MemberCheck result: true

RESULT: Desired result achieved. MemberCheck = true, business process progressed

Test Passed
```

Calculate initial payment (Deposit and membership fees) [service task] (form 1 service)

InitialCostCheck received variables: [InputVariable_1ndksl9, SignedUp, Breakdown, extraInfo, VehicleMake, CustomerName, VehicleModel, CustomerEmail, VehicleLocation, MembershipNumber, faultDescription] (tow selected)

TowRequest received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo, MemberCheck, VehicleMake, CustomerName, VehicleModel, customerName, CustomerEmail, customerEmail, depositAmount, towRequestSent, towingPriority, vehicleDetails, VehicleLocation, MembershipNumber, faultDescription, paymentTimestamp, breakdownLocation, towInfoAdditional, DescriptionOfFault, processInstanceKey, estimatedTowArrival, initialCostNotified, initialCostReceived, towRequestProcessed, towRequestTimestamp, initialCostTimestamp]

Tow Test

```
Activated 1 jobs for worker default and job type InitialCostCheck
------------------------------------------------------------------------------------------------
InitialCostCheck received variables: [InputVariable_1ndksl9, SignedUp, isMember, Breakdown,
SigningUp, extraInfo, MemberCheck, VehicleMake, CustomerName, VehicleModel, customerName,
CustomerEmail, customerEmail, depositAmount, towRequestSent, towingPriority, vehicleDetails,
VehicleLocation, MembershipNumber, faultDescription, paymentTimestamp, breakdownLocation,
towInfoAdditional, DescriptionOfFault, processInstanceKey, estimatedTowArrival, initialCostNotified,
initialCostReceived, towRequestProcessed, towRequestTimestamp, initialCostTimestamp]
------------------------------------------------------------------------------------------------
Form state - Existing member box checked: true, New member box checked: false
Membership number validated: 095208
Valid membership number confirmed: 095208
Calculated deposit: 150.00
MemberCheck result: true
Activated 1 jobs for worker default and job type inform-customer-init-cost
------------------------------------------------------------------------------------------------
inform-customer-init-cost received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo,
MemberCheck, VehicleMake, CustomerName, VehicleModel, customerName, CustomerEmail, customerEmail,
depositAmount, towRequestSent, towingPriority, vehicleDetails, VehicleLocation, MembershipNumber,
faultDescription, paymentTimestamp, breakdownLocation, towInfoAdditional, DescriptionOfFault,
processInstanceKey, estimatedTowArrival, initialCostNotified, initialCostReceived,
towRequestProcessed, towRequestTimestamp, initialCostTimestamp]
------------------------------------------------------------------------------------------------
Informing customer Enzo Joly of initial cost £150.00 for vehicle Honda Civic
Simulating payment receipt by sending message for process instance: 6755399457249505
Sending message 'ReceiveInitialCost' with correlation key '6755399457249505'
Message 'ReceiveInitialCost' sent successfully
------------------------------------------------------------------------------------------------
TowRequest received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo, MemberCheck,
VehicleMake, CustomerName, VehicleModel, customerName, CustomerEmail, customerEmail, depositAmount,
towRequestSent, towingPriority, vehicleDetails, VehicleLocation, MembershipNumber, faultDescription,
paymentTimestamp, breakdownLocation, towInfoAdditional, DescriptionOfFault, processInstanceKey,
estimatedTowArrival, initialCostNotified, initialCostReceived, towRequestProcessed,
towRequestTimestamp, initialCostTimestamp]
------------------------------------------------------------------------------------------------
Sending tow request to towing service
Breakdown location: Bristol
Vehicle details: Honda Civic - Brake Failure
Additional info: Nil
Activated 1 jobs for worker default and job type TowRequest
Sending message 'TowingRequest' with correlation key '6755399457249505'
Activated 1 jobs for worker default and job type process-tow-request
Message 'TowingRequest' sent successfully
------------------------------------------------------------------------------------------------
Received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo, MemberCheck, VehicleMake,
CustomerName, VehicleModel, customerName, CustomerEmail, customerEmail, depositAmount,
towRequestSent, towingPriority, vehicleDetails, VehicleLocation, MembershipNumber, faultDescription,
paymentTimestamp, breakdownLocation, towInfoAdditional, DescriptionOfFault, processInstanceKey,
estimatedTowArrival, initialCostNotified, initialCostReceived, towRequestProcessed,
towRequestTimestamp, initialCostTimestamp]
------------------------------------------------------------------------------------------------
```

```
Processing tow request for Honda Civic
Vehicle location: Bristol
Fault description: Brake Failure
Additional info: Nil
Membership status - existing: true, new: false, final: true
Tow request processed with priority: High
Vehicle details: Honda Civic - Brake Failure
Breakdown location: Bristol
```

RESULT: Desired result

Test Passed

No Tow Test

```
Activated 1 jobs for worker default and job type InitialCostCheck
-----------------------------------------------------------------------------------------------------------------
InitialCostCheck received variables: [InputVariable_1ndksl9, SignedUp, isMember, Breakdown,
SigningUp, extraInfo, MemberCheck, VehicleMake, CustomerName, VehicleModel, customerName,
CustomerEmail, customerEmail, depositAmount, towRequestSent, towingPriority, vehicleDetails,
VehicleLocation, MembershipNumber, faultDescription, paymentTimestamp, breakdownLocation,
towInfoAdditional, DescriptionOfFault, processInstanceKey, estimatedTowArrival, initialCostNotified,
initialCostReceived, towRequestProcessed, towRequestTimestamp, initialCostTimestamp]
-----------------------------------------------------------------------------------------------------------------
Form state - Existing member box checked: true, New member box checked: false
Membership number found in cache: 095208
Valid membership number confirmed: 095208
Calculated deposit: 150.00
MemberCheck result: true
Activated 1 jobs for worker default and job type inform-customer-init-cost
-----------------------------------------------------------------------------------------------------------------
inform-customer-init-cost received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo,
MemberCheck, VehicleMake, CustomerName, VehicleModel, customerName, CustomerEmail, customerEmail,
depositAmount, towRequestSent, towingPriority, vehicleDetails, VehicleLocation, MembershipNumber,
faultDescription, paymentTimestamp, breakdownLocation, towInfoAdditional, DescriptionOfFault,
processInstanceKey, estimatedTowArrival, initialCostNotified, initialCostReceived,
towRequestProcessed, towRequestTimestamp, initialCostTimestamp]
-----------------------------------------------------------------------------------------------------------------
Informing customer Enzo Joly of initial cost £150.00 for vehicle Honda Civic
Simulating payment receipt by sending message for process instance: 6755399457249505
Sending message 'ReceiveInitialCost' with correlation key '6755399457249505'
Message 'ReceiveInitialCost' sent successfully
```

RESULT: Desired Result

Test Passed

Receive initial costs [message receive task]

General Test (this part specifically)

```
Message 'TowingRequest' sent successfully
Received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo, MemberCheck, VehicleMake,
CustomerName, VehicleModel, customerName, CustomerEmail, customerEmail, depositAmount,
towRequestSent, towingPriority, vehicleDetails, VehicleLocation, MembershipNumber, faultDescription,
paymentTimestamp, breakdownLocation, towInfoAdditional, DescriptionOfFault, processInstanceKey,
estimatedTowArrival, initialCostNotified, initialCostReceived, towRequestProcessed,
towRequestTimestamp, initialCostTimestamp]
```

RESULT: Desired Result

Test Passed

Calculate cost of repairs [user form] [msg send]

General Test (successful message sending, variable consistency, application of discount)   MSG SEND

```
Activated 1 jobs for worker default and job type notify-reception-costing
Notifying reception of repair costs for Honda Civic: £333.00
Fault description: Brake Failure
```

```
CalculateFinalPrice received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo,
repairCost, MemberCheck, RepairCosts, VehicleMake, CustomerName, VehicleModel, customerName,
CustomerEmail, customerEmail, depositAmount, towRequestSent, towingPriority, vehicleDetails,
VehicleLocation, MembershipNumber, costingTimestamp, faultDescription, paymentTimestamp,
breakdownLocation, towInfoAdditional, DescriptionOfFault, processInstanceKey, estimatedTowArrival,
initialCostNotified, initialCostReceived, towRequestProcessed, towRequestTimestamp,
initialCostTimestamp, repairCostingNotified]
```

```
Original cost: 333.00
Is member: true
Discount: 10.00%
Final price: 299.70
Activated 1 jobs for worker default and job type CalculateFinalPrice
Activated 1 jobs for worker default and job type FinalQuote
Sending final quote to customer: Enzo Joly
Customer email: business@enzojoly.dev
Vehicle: Honda Civic
Final price: £299.70
Sending message 'QuoteNotification' with correlation key '6755399457249505'
Message 'QuoteNotification' sent successfully
```

```
RESULT: Desired result
```

`Test Passed`

Calculate membership discounts if applicable [service task]

```
Activated 1 jobs for worker default and job type CalculateFinalPrice
CalculateFinalPrice received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo,
quoteSent, TotalPrice, finalPrice, repairCost, MemberCheck, RepairCosts, VehicleMake, CustomerName,
VehicleModel, customerName, CustomerEmail, QuoteApproved, customerEmail, depositAmount,
quoteTimestamp, towRequestSent, towingPriority, vehicleDetails, VehicleLocation, discountApplied,
MembershipNumber, costingTimestamp, faultDescription, paymentTimestamp, QuoteApprovalForm,
breakdownLocation, towInfoAdditional, DescriptionOfFault, discountPercentage, processInstanceKey,
estimatedTowArrival, formattedFinalPrice, initialCostNotified, initialCostReceived,
towRequestProcessed, towRequestTimestamp, initialCostTimestamp, repairCostingNotified]
```

Is A Member Test

```
Original cost: 333.00
Is member: true
Discount: 10.00%
Final price: 299.70
Vehicle: Honda Civic
Final price: £299.70
Sending message 'QuoteNotification' with correlation key '6755399457249505'
Message 'QuoteNotification' sent successfully
```

```
RESULT: Desired result
```

`Test Passed`

Is Not A Member Test

```
Original cost: 333.00
Is member: false
Discount: 0.00%
Final price: 333.00
Activated 1 jobs for worker default and job type FinalQuote
Sending final quote to customer: Enzo Joly
```

```
Customer email: business@enzojoly.dev
Vehicle: Honda Civic
Final price: £333.00
Sending message 'QuoteNotification' with correlation key '6755399457249505'
Message 'QuoteNotification' sent successfully
```

RESULT: Desired result

Approve or deny the repair based on quote [user form]

Consent Given Test

```
Found QuoteApprovalForm with value: true
Processing customer approval: Approved
Sending message 'Approval' with correlation key '6755399457249505'
Message 'Approval' sent successfully
```

RESULT: Desired result

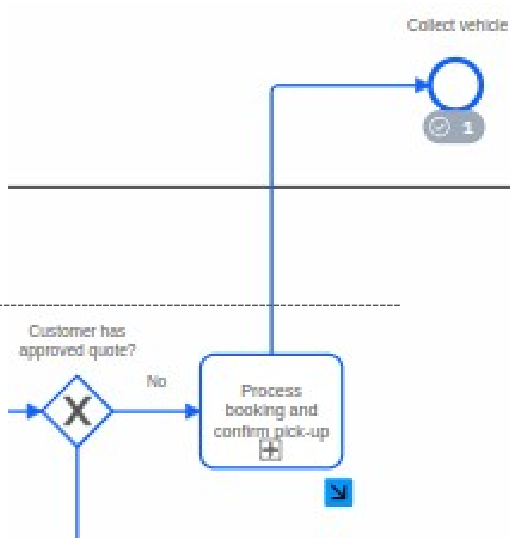
Test Passed

Consent NOT Given Test

```
Found QuoteApprovalForm with value: false
Processing customer approval: Denied
Sending message 'Approval' with correlation key '6755399457249505'
Activated 1 jobs for worker default and job type ArrangeCollection
Message 'Approval' sent successfully
```

RESULT: Desired result

Test Passed


Collect vehicle
1
Customer has approved quote?
No
Process booking and confirm pick-up

Process customer approval or denial [service task]

ABOVE Test (cont.)

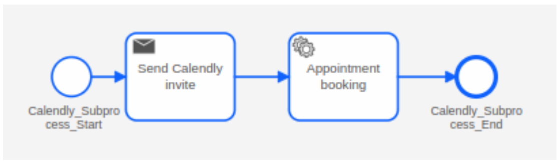
RESULT: Desired result

Test Passed

Process booking and confirm pick-up {SUB-PROCESS [service task] } working calendly link

We did not pay a premium Calendly subscription (outside the scope of the project) otherwise webhooks would be configured, instead we simulate a webhook. As such no “real” booking takes place but this functionality is assumed. Instead we chose Stripe API for our demonstration of external service integration.

Appointment Booking Test

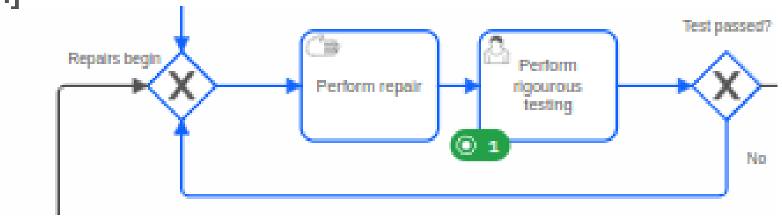


```
Sending message 'Approval' with correlation key '6755399457249505'
Activated 1 jobs for worker default and job type ArrangeCollection
Message 'Approval' sent successfully
Creating booking link for Enzo Joly - Vehicle collection
Generated booking URL: https://calendly.com/business-enzojoly/30min
email=business@enzojoly.dev&name=Enzo%20Joly&custom=Honda%20Civic&reason=Vehicle%20collection
Created Calendly booking link: https://calendly.com/business-enzojoly/30min
email=business@enzojoly.dev&name=Enzo%20Joly&custom=Honda%20Civic&reason=Vehicle%20collection
In production, an email would be sent to: business@enzojoly.dev
Vehicle details: Honda Civic
Sending message 'CollectionArranged' with correlation key '6755399457249505'
Activated 1 jobs for worker default and job type process-booking
Message 'CollectionArranged' sent successfully
No webhook data found, simulating booking confirmation
Simulated booking for Enzo Joly at 2025-03-27T10:00:00Z
Processing booking for Enzo Joly at 2025-03-27T10:00:00Z for vehicle: Honda Civic
```

RESULT: Desired result

Test Passed

Perform rigorous testing [user form]



Tests NOT Passing Test

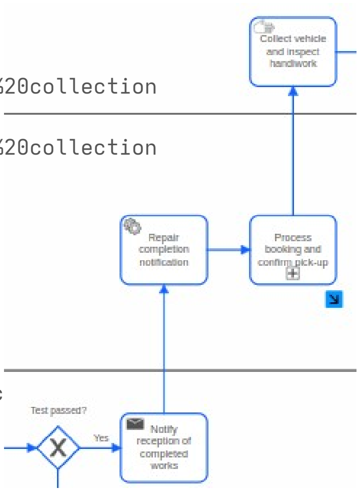
```
[No terminal output but BPMN looped]
```

RESULT: Desired result

Test Passed

Tests Confirmed Passing Test

```
Activated 1 jobs for worker default and job type NotifyWorkComplete
Notifying reception that vehicle repairs are complete
Vehicle: Honda Civic
Sending message 'WorksComplete' with correlation key '6755399457249505'
Activated 1 jobs for worker default and job type repair-complete-notify
Message 'WorksComplete' sent successfully
Processing repair completion notification for the receptionist
Repair completion details being recorded for customer notification
Sending message 'WorksComplete' with correlation key '6755399457249505'
Message 'WorksComplete' sent successfully
Creating booking link for Enzo Joly - Vehicle collection
Generated booking URL: https://calendly.com/business-enzojoly/30min
email=business@enzojoly.dev&name=Enzo%20Joly&custom=Honda%20Civic&reason=Vehicle%20collection
Created Calendly booking link: https://calendly.com/business-enzojoly/30min
email=business@enzojoly.dev&name=Enzo%20Joly&custom=Honda%20Civic&reason=Vehicle%20collection
In production, an email would be sent to: business@enzojoly.dev
Vehicle details: Honda Civic
Activated 1 jobs for worker default and job type ArrangeCollection
Sending message 'CollectionArranged' with correlation key '6755399457249505'
Activated 1 jobs for worker default and job type process-booking
Message 'CollectionArranged' sent successfully
Using existing booking info from webhook
Processing booking for Enzo Joly at 2025-03-27T10:00:00Z for vehicle: Honda Civic
```


Collect vehicle and inspect handiwork
Repair completion notification
Process booking and confirm pick-up
Test passed?
Yes
Notify reception of completed works

RESULT: Desired result. Sub-process triggered. BPMN progressed

Test Passed

Repair completion notification [service task]

ABOVE Test

```
Activated 1 jobs for worker default and job type repair-complete-notify
Message 'WorksComplete' sent successfully
```

RESULT: Desired result. Message sent

Test Passed

Verify customer satisfaction [user form]

Customer NOT Satisfied Test

```
Activated 1 jobs for worker default and job type process-satisfaction
Found CustomerSatisfactionForm with value: NotSatisfied
Customer satisfaction status determined as: Not Satisfied
Setting CustomerSatisfied to: false
```

RESULT: Desired result

Test Passed

Customer Satisfied Test

```
Activated 1 jobs for worker default and job type process-satisfaction
Found CustomerSatisfactionForm with value: Satisfied
Customer satisfaction status determined as: Satisfied
Setting CustomerSatisfied to: true
```

```
RESULT: Desired result. Stripe processing triggered
```

```
Test Passed
```

Process customer satisfaction [service task]

Above Test

```
process-approval received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo,
quoteSent, TotalPrice, finalPrice, repairCost, MemberCheck, RepairCosts, TestsPassed, VehicleMake,
bookingLink, CustomerName, VehicleModel, customerName, CustomerEmail, QuoteApproved, customerEmail,
depositAmount, quoteTimestamp, towRequestSent, towingPriority, vehicleDetails, VehicleLocation,
appointmentTime, discountApplied, MembershipNumber, bookingConfirmed, bookingReference,
costingTimestamp, faultDescription, notificationSent, paymentTimestamp, textfield_cny2ct,
CustomerSatisfied, QuoteApprovalForm, breakdownLocation, customerSatisfied, towInfoAdditional,
DescriptionOfFault, appointmentEndTime, discountPercentage, processInstanceKey, completionTimestamp,
estimatedTowArrival, formattedFinalPrice, initialCostNotified, initialCostReceived,
towRequestProcessed, towRequestTimestamp, initialCostTimestamp, repairCostingNotified,
CustomerSatisfactionForm, repairCompletionProcessed, repairCompletionTimestamp,
workCompleteNotificationSent]
```

```
RESULT: Desired result
```

Generate Stripe invoice [service task]

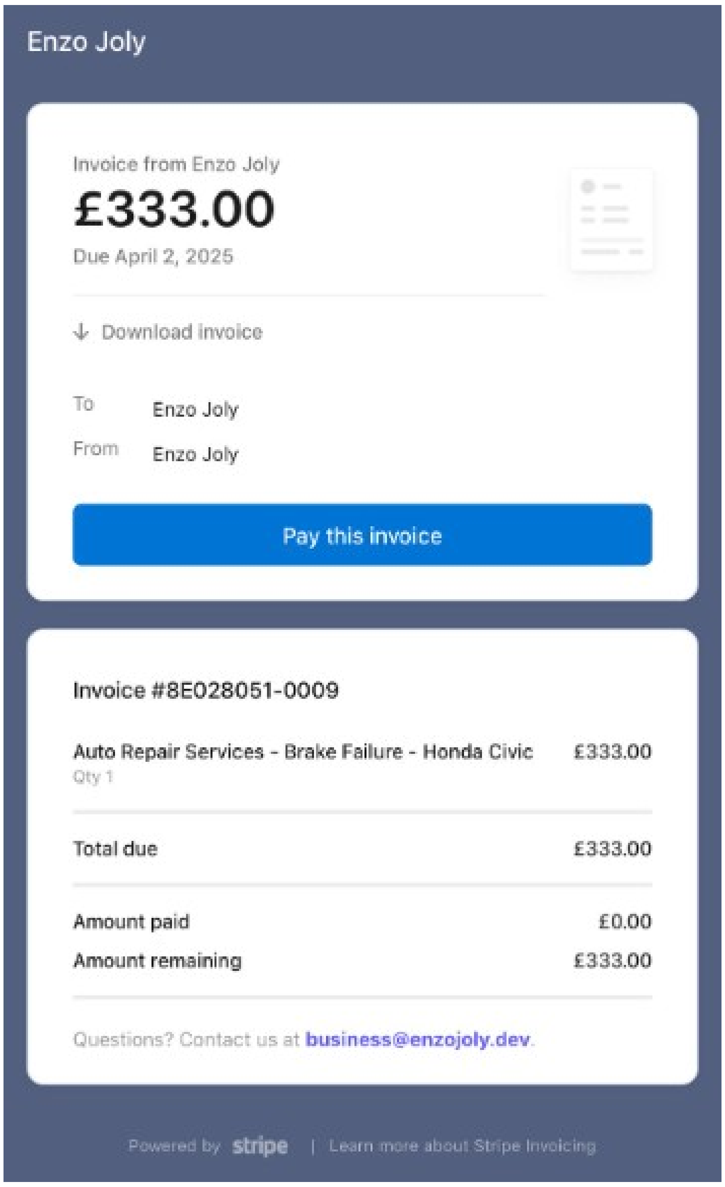
Invoice Generation and Sending Test

```
stripe-invoice worker received variables: [SignedUp, isMember, Breakdown, SigningUp, extraInfo,
quoteSent, TotalPrice, finalPrice, repairCost, MemberCheck, RepairCosts, TestsPassed, VehicleMake,
bookingLink, CustomerName, VehicleModel, customerName, CustomerEmail, QuoteApproved, customerEmail,
depositAmount, quoteTimestamp, towRequestSent, towingPriority, vehicleDetails, VehicleLocation,
appointmentTime, discountApplied, MembershipNumber, bookingConfirmed, bookingReference,
costingTimestamp, faultDescription, notificationSent, paymentTimestamp, textfield_cny2ct,
CustomerSatisfied, QuoteApprovalForm, breakdownLocation, customerSatisfied, towInfoAdditional,
DescriptionOfFault, appointmentEndTime, discountPercentage, processInstanceKey, completionTimestamp,
estimatedTowArrival, formattedFinalPrice, initialCostNotified, initialCostReceived,
towRequestProcessed, towRequestTimestamp, initialCostTimestamp, repairCostingNotified,
CustomerSatisfactionForm, repairCompletionProcessed, repairCompletionTimestamp,
workCompleteNotificationSent]
Generating invoice for customer: Enzo Joly
Vehicle: Honda Civic
Amount: £333.00
Using Stripe in PRODUCTION mode
Generating Stripe invoice for customer: Enzo Joly
Invoice amount: £333.00
Using Stripe API in PRODUCTION mode
Creating/retrieving customer with email: business@enzojoly.dev, name: Enzo Joly
Activated 1 jobs for worker default and job type stripe-invoice
Calling Stripe API: /customers with data: email=business@enzojoly.dev, name=Enzo Joly,
Stripe API response status: 200
Stripe API response body: { . . . }
```
[very extensive api call body omitted]

```
Using customer with ID: cus_S3PHZ173ZOz8SC
Amount before conversion: 333.0
Amount after conversion to smallest unit: 33300
Invoice item data being sent to Stripe: {"customer":"cus_S3PHZ173ZOz8SC","description":"Auto Repair
Services - Brake Failure - Honda Civic","amount":33300,"currency":"gbp"}
--------------------------------------------------------------------------------
Created invoice with ID: in_1R9IT9K76YVOCfmrHjm1qkPT
Invoice initial amount_due: 33300
Finalizing invoice with ID: in_1R9IT9K76YVOCfmrHjm1qkPT
Calling Stripe API: /invoices/in_1R9IT9K76YVOCfmrHjm1qkPT/finalize with data:
Stripe API response status: 200
Stripe API response body: { . . . }
--------------------------------------------------------------------------------
Invoice email sent successfully to customer
Successfully created Stripe invoice with ID: in_1R9IT9K76YVOCfmrHjm1qkPT
Sending message 'InvoiceGenerated' with correlation key '67553994572495O5'
Message 'InvoiceGenerated' sent successfully
Invoice generation completed successfully. Invoice ID: in_1R9IT9K76YVOCfmrHjm1qkPT
```

```
RESULT: Desired result
```

```
Test Passed
```



Enzo Joly

Invoice from Enzo Joly

£333.00

Due April 2, 2025

Download invoice

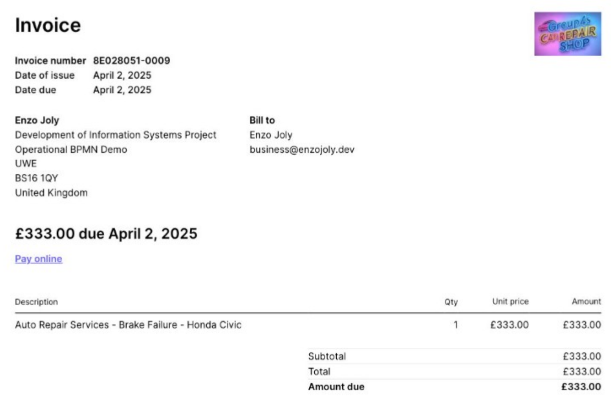
To	Enzo Joly
From	Enzo Joly

Invoice #8E028051-0009

Auto Repair Services - Brake Failure - Honda Civic Qty 1	£333.00
Total due	£333.00
Amount paid	£0.00
Amount remaining	£333.00

Questions? Contact us at business@enzojoly.dev.

Powered by stripe | Learn more about Stripe Invoicing



Invoice

Invoice number	**8E028051-0009**
Date of issue	April 2, 2025
Date due	April 2, 2025

Enzo Joly
Development of Information Systems Project
Operational BPMN Demo
UWE
BS16 1QY
United Kingdom

Bill to
Enzo Joly
business@enzojoly.dev

£333.00 due April 2, 2025

Pay online

Description	Qty	Unit price	Amount
Auto Repair Services - Brake Failure - Honda Civic	1	£333.00	£333.00
		Subtotal	£333.00
		Total	£333.00
		Amount due	**£333.00**