Portfolio: Systems Development Portfolio

IT Support Ticketing System Implementation

Writing Report

Project Manager/Leader: Rayan Louahche

Group Members: Rayan Louahche

Development of Information Systems Projects (DISP)
UFCFAF-30-3

Final Word Count:2000words

Date of submission: 25/04/24

Table of Contents:

Executive Summary:

The Project report outlines the overall steps, objectives and milestones taken through the design, development and management of the ticketing System asked to conceive by the client Mr Daniel Hatherall and supervised by Mr Dilshan Jayatilake.

Project Management:

Project Plan:

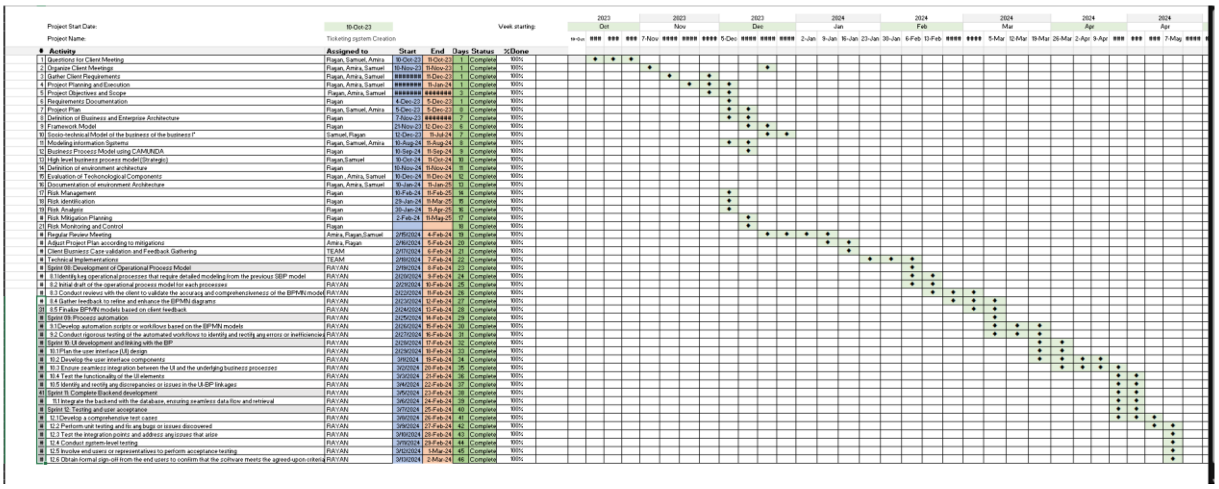
The Project Plan includes evidence of effective project management, agreed requirements, detailed timeline, resource allocation and risk mitigation strategies that guided the project execution and ensured constant alignment with the business objectives and stakeholder expectations.

Solo development Methodology:

Given the solo nature of the project, an Individual Development Methodology was adopted, necessitating a heightened level of self-reliance and multitasking. The solo development methodology was employed to manage the project effectively. This approach involved setting clear and precise objectives, prioritizing tasks based on their impact and dependencies and leveraging iterative development cycles to validate and refine the solution incrementally.

Setting Clear Objectives: Initial stages involved defining clear project objectives and deliverables. This was critical in maintaining focus and direction, particularly in a solo development environment where the risk of scope creep is high.

Find all clear objectives the following Gantt Chart that was deployed to assist and organize this project realisation: *(Zoom 420% for clear reading)*



Project Start Date: 10-Oct-23
Project Name: Ticketing system Creation

#	Activity	Assigned to	Start	End	Days	Status	%Done
1	Questions for Client Meeting	Rayan, Samuel, Amira	10-Oct-23	11-Oct-23	1	Complete	100%
2	Organize Client Meetings	Rayan, Amira, Samuel	10-Nov-23	11-Nov-23	1	Complete	100%
3	Gather Client Requirements	Rayan, Amira, Samuel	#######	11-Dec-23	1	Complete	100%
4	Project Planning and Execution	Rayan, Amira, Samuel	#######	11-Jan-24	1	Complete	100%
5	Project Objectives and Scope	Rayan, Amira, Samuel	#######	#######	3	Complete	100%
6	Requirements Documentation	Rayan	4-Dec-23	5-Dec-23	1	Complete	100%
7	Project Plan	Rayan, Samuel, Amira	5-Dec-23	5-Dec-23	0	Complete	100%
8	Definition of Business and Enterprise Architecture	Rayan	7-Nov-23	#######	7	Complete	100%
9	Framework Model	Rayan	21-Nov-23	12-Dec-23	6	Complete	100%
10	Socio-technical Model of the business of the business I"	Samuel, Rayan	12-Dec-23	11-Jul-24	7	Complete	100%
11	Modeling information Systems	Rayan, Samuel, Amira	10-Aug-24	11-Aug-24	8	Complete	100%
12	Business Process Model using CAMUNDA	Rayan	10-Sep-24	11-Sep-24	9	Complete	100%
13	High level business process model (Strategic)	Rayan,Samuel	10-Oct-24	11-Oct-24	10	Complete	100%
14	Definition of environment architecture	Rayan	10-Nov-24	11-Nov-24	11	Complete	100%
15	Evaluation of Techonological Components	Rayan , Amira, Samuel	10-Dec-24	11-Dec-24	12	Complete	100%
16	Documentation of environment Architecture	Rayan, Amira, Samuel	10-Jan-24	11-Jan-25	13	Complete	100%
17	Risk Management	Rayan	10-Feb-24	11-Feb-25	14	Complete	100%
18	Risk identification	Rayan	29-Jan-24	11-Mar-25	15	Complete	100%
19	Risk Analysis	Rayan	30-Jan-24	11-Apr-25	16	Complete	100%
#	Risk Mitigation Planning	Rayan	2-Feb-24	11-May-25	17	Complete	100%
21	Risk Monitoring and Control	Rayan			18	Complete	100%
#	Regular Review Meeting	Amira, Rayan,Samuel	2/15/2024	4-Feb-24	19	Complete	100%
#	Adjust Project Plan according to mitigations	Amira, Rayan	2/16/2024	5-Feb-24	20	Complete	100%
#	Client Busniess Case validation and Feedback Gathering	TEAM	2/17/2024	6-Feb-24	21	Complete	100%
#	Technical Implementations	TEAM	2/18/2024	7-Feb-24	22	Complete	100%
#	Sprint 08: Development of Operational Process Model	RAYAN	2/19/2024	8-Feb-24	23	Complete	100%
#	8.1 Identify key operational processes that require detailed modeling from the previous SBP model	RAYAN	2/20/2024	9-Feb-24	24	Complete	100%
#	8.2 Initial draft of the operational process model for each processes	RAYAN	2/21/2024	10-Feb-24	25	Complete	100%
#	8.3 Conduct reviews with the client to validate the accuracy and comprehensiveness of the BPMN model	RAYAN	2/22/2024	11-Feb-24	26	Complete	100%
#	8.4 Gather feedback to refine and enhance the BPMN diagrams	RAYAN	2/23/2024	12-Feb-24	27	Complete	100%
31	8.5 Finalize BPMN models based on client feedback	RAYAN	2/24/2024	13-Feb-24	28	Complete	100%
#	Sprint 09: Process automation	RAYAN	2/25/2024	14-Feb-24	29	Complete	100%
#	9.1 Develop automation scripts or workflows based on the BPMN models	RAYAN	2/26/2024	15-Feb-24	30	Complete	100%
#	9.2 Conduct rigorous testing of the automated workflows to identify and rectify any errors or inefficiencies	RAYAN	2/27/2024	16-Feb-24	31	Complete	100%
#	Sprint 10: UI development and linking with the BP	RAYAN	2/28/2024	17-Feb-24	32	Complete	100%
#	10.1 Plan the user interface (UI) design	RAYAN	2/29/2024	18-Feb-24	33	Complete	100%
#	10.2 Develop the user interface components	RAYAN	3/1/2024	19-Feb-24	34	Complete	100%
#	10.3 Ensure seamless integration between the UI and the underlying business processes	RAYAN	3/2/2024	20-Feb-24	35	Complete	100%
#	10.4 Test the functionality of the UI elements	RAYAN	3/3/2024	21-Feb-24	36	Complete	100%
#	10.5 Identify and rectify any discrepancies or issues in the UI-BP linkages	RAYAN	3/4/2024	22-Feb-24	37	Complete	100%
41	Sprint 11: Complete Backend development	RAYAN	3/5/2024	23-Feb-24	38	Complete	100%
#	11.1 Integrate the backend with the database, ensuring seamless data flow and retrieval	RAYAN	3/6/2024	24-Feb-24	39	Complete	100%
#	Sprint 12: Testing and user acceptance	RAYAN	3/7/2024	25-Feb-24	40	Complete	100%
#	12.1 Develop a comprehensive test cases	RAYAN	3/8/2024	26-Feb-24	41	Complete	100%
#	12.2 Perform unit testing and fix any bugs or issues discovered	RAYAN	3/9/2024	27-Feb-24	42	Complete	100%
#	12.3 Test the integration points and address any issues that arise	RAYAN	3/10/2024	28-Feb-24	43	Complete	100%
#	12.4 Conduct system-level testing	RAYAN	3/11/2024	29-Feb-24	44	Complete	100%
#	12.5 Involve end users or representatives to perform acceptance testing	RAYAN	3/12/2024	1-Mar-24	45	Complete	100%
#	12.6 Obtain formal sign-off from the end users to confirm that the software meets the agreed-upon criteria	RAYAN	3/13/2024	2-Mar-24	46	Complete	100%

Self-Education and Skill Development: A significant amount of time was devoted to self-education in Camunda BPMN. This involved extensive research on Camunda forums, comprehensive learning and practice using tools like Camunda Modeler, Camunda Cloud, Zeebe client, and Docker Desktop using a Linux Virtual Machine. Proficiency in SMTP and HTTP request handling was achieved, along with mastering Postman for calling correlation keys and handling event-based gateways and message intermediate catch events. This intensive learning phase was crucial but also posed challenges in terms of time management and prioritization.

Tools, Software and Technologies: Overview of all the tools and technologies used during the realisation of the project:

Camunda 7/Camunda Modeler/Camunda Cockpit: Used for training, designing, modelling, and monitoring BPMN workflows.

Camunda 8/Camunda Cloud: Leveraged for cloud-based process automation and deployment.

Zeebe Client: Employed to interact with Zeebe workflows and orchestrate microservices.

Docker Desktop: Used for testing, containerization to ensure consistency across different environments.

Linux VM: Utilized to establish separate environments during tests but also as a development environment to host and run testing the application.

IntelliJ IDE and VScode: Chosen for Java development to enhance productivity and code quality.

MongoDB Shell + MongoDB Compass: Used for storing tickets, Survey's inputs and database management and querying.

JavaMail sender: Integrated to handle email notifications and communications.

Languages of Coding Used:

Java: Implemented for service tasks and backend development.

JSON: Utilized for NoSQL database operations, configuration and creating Camunda Embedded forms

HTML: Developed for creating surveys and some user interfaces.

XML: Employed for defining BPMN processes and workflows.

Iterative Development and Testing: Adopting an iterative approach was essential for incremental development and frequent testing. This iterative process allowed for continuous feedback and refinement, although managing multiple iterations single-handedly posed challenges in maintaining consistency and tracking changes effectively.

Proactive Risk Management: A risk mitigation plan was developed to identify potential risks and corresponding mitigation strategies. Regular reviews and updates were conducted to adapt to evolving challenges, ensuring proactive risk management throughout the project lifecycle.

Quality and Configuration Management:

Code Review: Throughout the design and build up of the process self-code reviews have been conducted every single day to maintain a stable and consistent working environment.

(Screenshots of code)

Development Environment:

Configured a Linux VM on Docker Desktop for a consistent and isolated development environment.

Database Setup: Deployed MongoDB using Docker containers to manage and store ticket and survey data.

Integration Environment:

Integrated Camunda Cloud and Zeebe Client for workflow orchestration and microservices integration.

Testing Environment:

Established separate environments for unit testing, integration testing, and user acceptance testing to ensure thorough and systematic testing.

Test Plan:

Environment Setup: Describe the test environment setup and any challenges faced during this process.

A comprehensive test plan was devised to uphold the quality and reliability of the Ticketing System. It encompassed unit tests, integration tests, and user acceptance tests. However, the solo nature of the project presented challenges in comprehensive testing, with limitations in terms of resources and diverse perspectives for testing:

Objective: Ensure the quality, reliability, and functionality of the Ticketing System developed using the Camunda BPM platform.

1) Unit and Integration Testing:

Objective 1: Validate the individual components or units of the Ticketing System for correctness.

Objective 2: Validate the interaction between integrated components of the Ticketing System.

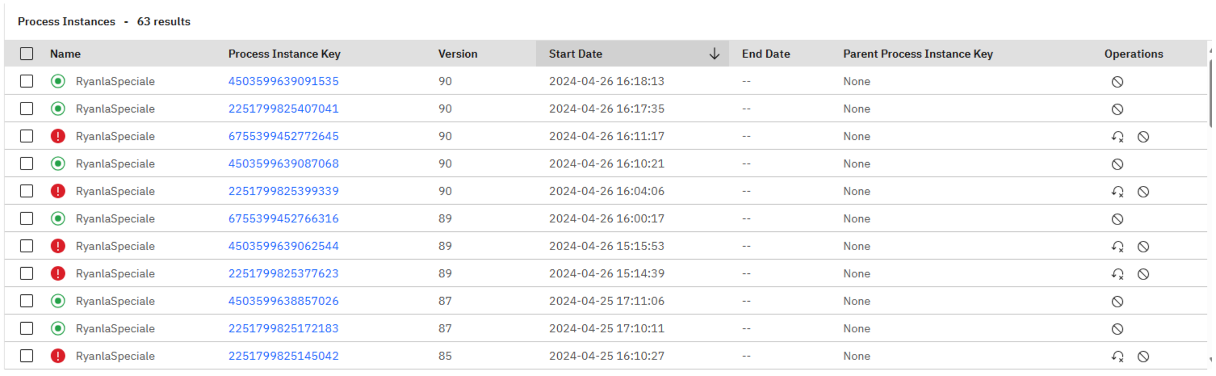
Test ID	Test Description	Expected Outcome	Number of Tries until Success	Status
UT-001	Testing if User can choose whether to create a ticket whether by email or web	Sequence flow working and get passed to either (Enter Email Details) User Task or (Create ticket) user task	10	PASS

Test ID	Test Description	Expected Outcome	Number of Tries until Success	Status
UT-002	Creating a JSON configuration form to send a ticket by email	Form is supposed to store the User’s Input before sending an email using the (Send Email) worker	8	PASS
UT-003	Creating a Camunda Linked form to send a ticket by web	Form is supposed to store the User’s input when sending a ticket using web	1	PASS
UT-004	Testing if the ticket is created using the Web	Ticket created via Web	5	PASS
UT-005	Testing if my Email Worker is running and working	Supposed to receive a test email to a personal Gmail account by when filling the User task’s form (Enter Email Details)	50 times or over (Spent a whole day on it)	PASS
UT-006	Testing if the user can create a ticket using email	Ticket created via email	X	PASS
UT-007	Testing if the email ticket is reached to the IT System and is sent	Email is getting passed properly to the IT-System and worker is sending SMTP correctly	X	PASS
UT-008	Testing if invalid Email Error Boundary event throws an error and brings back to the ticket creation	Throw an error in an invalid Email address is inserted and brings back the process to entering the (Email Details) user task	2	PASS
UT-009	Testing if my MongoDB Job Handler is working	After ticket is created either by email of web, the ticket’s data is saved in a NoSQL DB	Around 20	PASS
UT-010	Creating Form for all other user task	Forms supposed to be linked to my BPMN after deployment, Form’s Input should be stored and considered.	1	PASS
UT-011	Testing loops	After a node is triggered two times, it should let the process continue and consider the new data inserted by the user	33 or even more	PASS
UT-012	Testing if Worker for the request (additional info by email) service task,	After loop is triggered the service task should trigger the worker to send an email to the use asking for more	4	PASS

Test ID	Test Description	Expected Outcome	Number of Tries until Success	Status
	sends an email back to the User to ask for additional info	information concerning his enquiry for the resolution of his ticket		
UT-013	Sequence flows	Link every User task's form the linked sequence flows using Boolean conditions for example: if user input the Checkbox Yes then the instance passes through to the sequence flow that has as a condition the checkbox ID = Yes and vice versa	30, took some time to find the right function as I had to investigate the forums	PASS
UT-014	Other Workers and Job Handlers	Workers are linked to each Service tasks using the task definition, and each service task such as switching the ticket status from open to close to complete should work perfectly fine and be linked	Over 50, some service tasks are quite far in the BPMN process which makes it difficult to make proficient testing	PASS
UT-015	HTML survey Testing	The survey created using HTML is professional and stores the data correctly, Survey's input is linked into the mongo DB database in a new collection called Survey	5	PASS
UT-016	Testing if survey data is passed to the MongoDB database	The data inserted in the survey is store the TicketDB database, inside of the collection survey	9	PASS
UT-017	Testing if the send survey service task works	Service task is supposed to send the HTML survey using Email send the HTML survey using Email to the User	17	PASS
UT-018	Testing Intermediate Catch Event	Intermeddiate Catch Event is supposed to catch whether the user opened and filled the survey or ignored it and didn't fill it	13	PASS

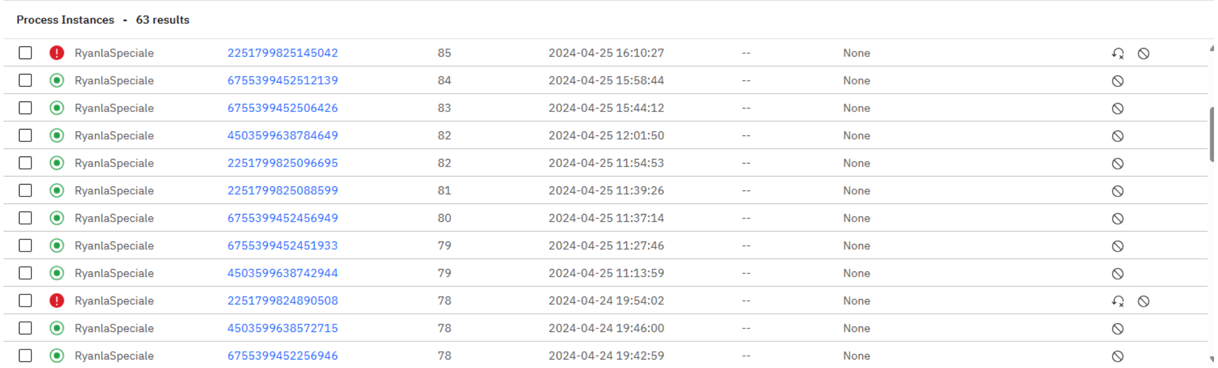
Test ID	Test Description	Expected Outcome	Number of Tries until Success	Status
UT-019	Testing Message Intermediate Catch Event	Message intermediate catch event triggers when the user opens the survey	40 (Spent a whole day on it until a Camunda dev tell me its not possible to make it work)	FAIL
UT-020	Testing Timer Intermediate catch event	Timer Intermediate catch event automatically triggers after the user didn't open the survey for 10 seconds (10secs value inserted only for testing of purposes)	2	PASS
UT-021	Testing process termination	End Event Triggered and Instance finished Successfully	Accumulation of all testings combined	PASS

Overall deployed Testing Instances before requirements met with efficiency and final Product Delivered:



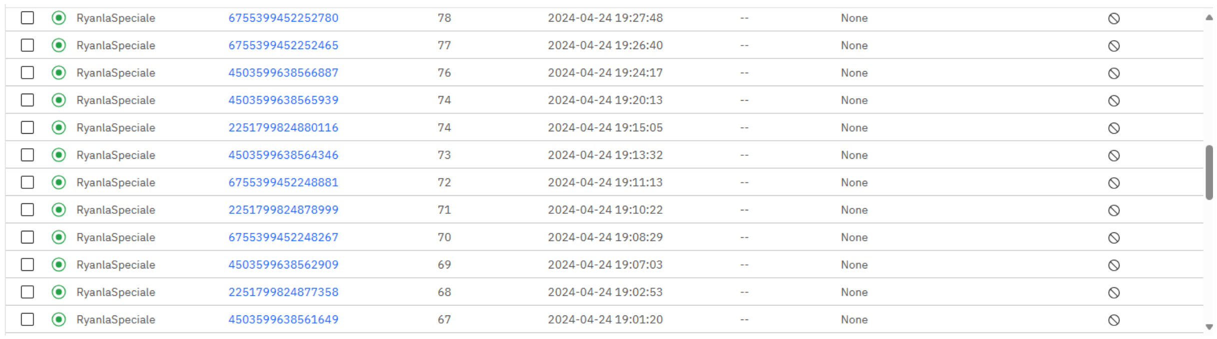
Process Instances - 63 results

	Name	Process Instance Key	Version	Start Date ↓	End Date	Parent Process Instance Key	Operations
●	RyanlaSpeciale	4503599639091535	90	2024-04-26 16:18:13	--	None	⊘
●	RyanlaSpeciale	2251799825407041	90	2024-04-26 16:17:35	--	None	⊘
❗	RyanlaSpeciale	6755399452772645	90	2024-04-26 16:11:17	--	None	↺ ⊘
●	RyanlaSpeciale	4503599639087068	90	2024-04-26 16:10:21	--	None	⊘
❗	RyanlaSpeciale	2251799825399339	90	2024-04-26 16:04:06	--	None	↺ ⊘
●	RyanlaSpeciale	6755399452766316	89	2024-04-26 16:00:17	--	None	⊘
❗	RyanlaSpeciale	4503599639062544	89	2024-04-26 15:15:53	--	None	↺ ⊘
❗	RyanlaSpeciale	2251799825377623	89	2024-04-26 15:14:39	--	None	↺ ⊘
●	RyanlaSpeciale	4503599638857026	87	2024-04-25 17:11:06	--	None	⊘
●	RyanlaSpeciale	2251799825172183	87	2024-04-25 17:10:11	--	None	⊘
❗	RyanlaSpeciale	2251799825145042	85	2024-04-25 16:10:27	--	None	↺ ⊘

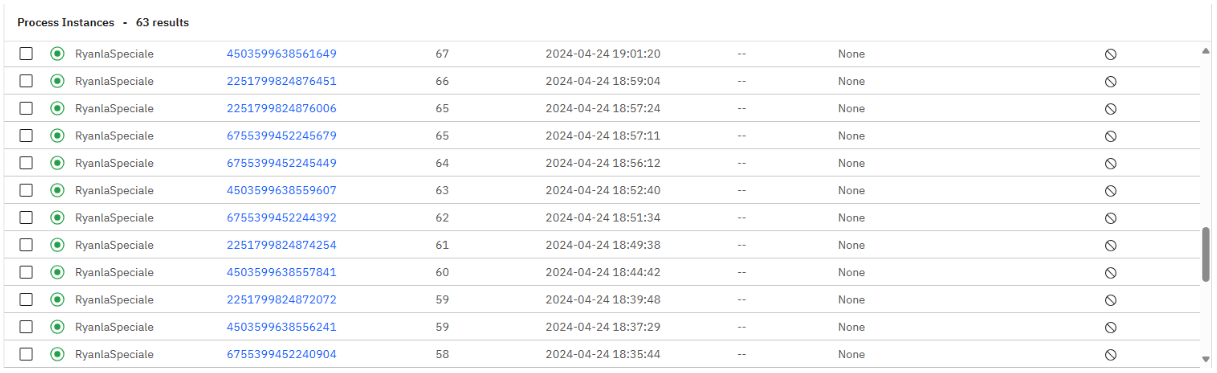


Process Instances - 63 results

❗	RyanlaSpeciale	2251799825145042	85	2024-04-25 16:10:27	--	None	↺ ⊘
●	RyanlaSpeciale	6755399452512139	84	2024-04-25 15:58:44	--	None	⊘
●	RyanlaSpeciale	6755399452506426	83	2024-04-25 15:44:12	--	None	⊘
●	RyanlaSpeciale	4503599638784649	82	2024-04-25 12:01:50	--	None	⊘
●	RyanlaSpeciale	2251799825096695	82	2024-04-25 11:54:53	--	None	⊘
●	RyanlaSpeciale	2251799825088599	81	2024-04-25 11:39:26	--	None	⊘
●	RyanlaSpeciale	6755399452456949	80	2024-04-25 11:37:14	--	None	⊘
●	RyanlaSpeciale	6755399452451933	79	2024-04-25 11:27:46	--	None	⊘
●	RyanlaSpeciale	4503599638742944	79	2024-04-25 11:13:59	--	None	⊘
❗	RyanlaSpeciale	2251799824890508	78	2024-04-24 19:54:02	--	None	↺ ⊘
●	RyanlaSpeciale	4503599638572715	78	2024-04-24 19:46:00	--	None	⊘
●	RyanlaSpeciale	6755399452256946	78	2024-04-24 19:42:59	--	None	⊘

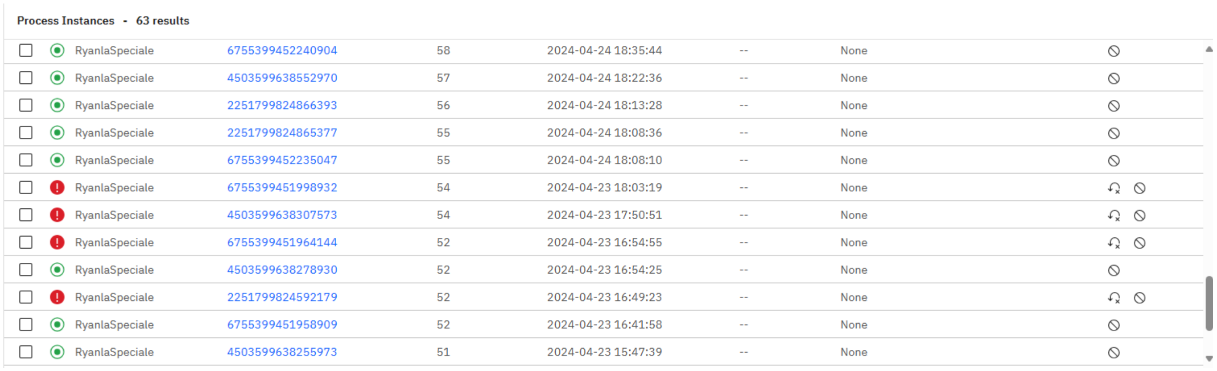


☐	●	RyanlaSpeciale	6755399452252780	78	2024-04-24 19:27:48	--	None	⦸
☐	●	RyanlaSpeciale	6755399452252465	77	2024-04-24 19:26:40	--	None	⦸
☐	●	RyanlaSpeciale	4503599638566887	76	2024-04-24 19:24:17	--	None	⦸
☐	●	RyanlaSpeciale	4503599638565939	74	2024-04-24 19:20:13	--	None	⦸
☐	●	RyanlaSpeciale	2251799824880116	74	2024-04-24 19:15:05	--	None	⦸
☐	●	RyanlaSpeciale	4503599638564346	73	2024-04-24 19:13:32	--	None	⦸
☐	●	RyanlaSpeciale	6755399452248881	72	2024-04-24 19:11:13	--	None	⦸
☐	●	RyanlaSpeciale	2251799824878999	71	2024-04-24 19:10:22	--	None	⦸
☐	●	RyanlaSpeciale	6755399452248267	70	2024-04-24 19:08:29	--	None	⦸
☐	●	RyanlaSpeciale	4503599638562909	69	2024-04-24 19:07:03	--	None	⦸
☐	●	RyanlaSpeciale	2251799824877358	68	2024-04-24 19:02:53	--	None	⦸
☐	●	RyanlaSpeciale	4503599638561649	67	2024-04-24 19:01:20	--	None	⦸



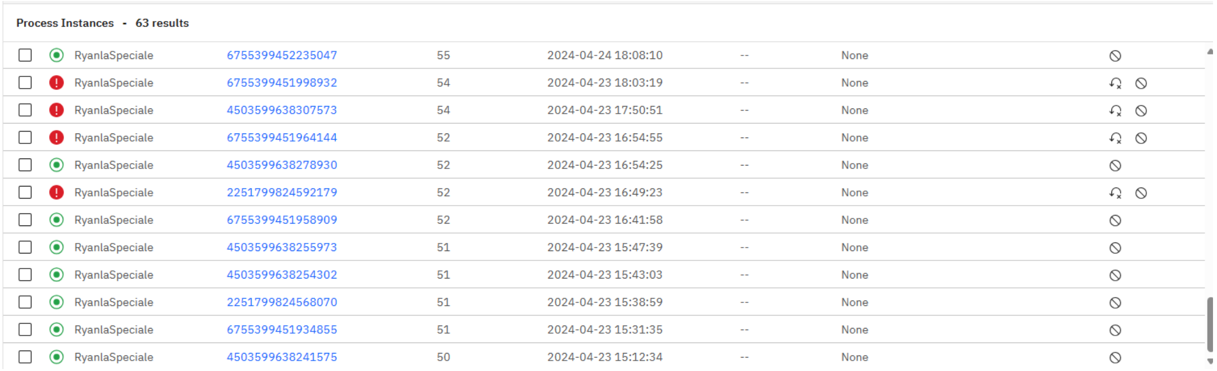
Process Instances - 63 results

☐	●	RyanlaSpeciale	4503599638561649	67	2024-04-24 19:01:20	--	None	⦸
☐	●	RyanlaSpeciale	2251799824876451	66	2024-04-24 18:59:04	--	None	⦸
☐	●	RyanlaSpeciale	2251799824876006	65	2024-04-24 18:57:24	--	None	⦸
☐	●	RyanlaSpeciale	6755399452245679	65	2024-04-24 18:57:11	--	None	⦸
☐	●	RyanlaSpeciale	6755399452245449	64	2024-04-24 18:56:12	--	None	⦸
☐	●	RyanlaSpeciale	4503599638559607	63	2024-04-24 18:52:40	--	None	⦸
☐	●	RyanlaSpeciale	6755399452244392	62	2024-04-24 18:51:34	--	None	⦸
☐	●	RyanlaSpeciale	2251799824874254	61	2024-04-24 18:49:38	--	None	⦸
☐	●	RyanlaSpeciale	4503599638557841	60	2024-04-24 18:44:42	--	None	⦸
☐	●	RyanlaSpeciale	2251799824872072	59	2024-04-24 18:39:48	--	None	⦸
☐	●	RyanlaSpeciale	4503599638556241	59	2024-04-24 18:37:29	--	None	⦸
☐	●	RyanlaSpeciale	6755399452240904	58	2024-04-24 18:35:44	--	None	⦸



Process Instances - 63 results

☐	●	RyanlaSpeciale	6755399452240904	58	2024-04-24 18:35:44	--	None	⦸
☐	●	RyanlaSpeciale	4503599638552970	57	2024-04-24 18:22:36	--	None	⦸
☐	●	RyanlaSpeciale	2251799824866393	56	2024-04-24 18:13:28	--	None	⦸
☐	●	RyanlaSpeciale	2251799824865377	55	2024-04-24 18:08:36	--	None	⦸
☐	●	RyanlaSpeciale	6755399452235047	55	2024-04-24 18:08:10	--	None	⦸
☐	❗	RyanlaSpeciale	6755399451998932	54	2024-04-23 18:03:19	--	None	↻ ⦸
☐	❗	RyanlaSpeciale	4503599638307573	54	2024-04-23 17:50:51	--	None	↻ ⦸
☐	❗	RyanlaSpeciale	6755399451964144	52	2024-04-23 16:54:55	--	None	↻ ⦸
☐	●	RyanlaSpeciale	4503599638278930	52	2024-04-23 16:54:25	--	None	⦸
☐	❗	RyanlaSpeciale	2251799824592179	52	2024-04-23 16:49:23	--	None	↻ ⦸
☐	●	RyanlaSpeciale	6755399451958909	52	2024-04-23 16:41:58	--	None	⦸
☐	●	RyanlaSpeciale	4503599638255973	51	2024-04-23 15:47:39	--	None	⦸



Process Instances - 63 results

☐	●	RyanlaSpeciale	6755399452235047	55	2024-04-24 18:08:10	--	None	⦸
☐	❗	RyanlaSpeciale	6755399451998932	54	2024-04-23 18:03:19	--	None	↻ ⦸
☐	❗	RyanlaSpeciale	4503599638307573	54	2024-04-23 17:50:51	--	None	↻ ⦸
☐	❗	RyanlaSpeciale	6755399451964144	52	2024-04-23 16:54:55	--	None	↻ ⦸
☐	●	RyanlaSpeciale	4503599638278930	52	2024-04-23 16:54:25	--	None	⦸
☐	❗	RyanlaSpeciale	2251799824592179	52	2024-04-23 16:49:23	--	None	↻ ⦸
☐	●	RyanlaSpeciale	6755399451958909	52	2024-04-23 16:41:58	--	None	⦸
☐	●	RyanlaSpeciale	4503599638255973	51	2024-04-23 15:47:39	--	None	⦸
☐	●	RyanlaSpeciale	4503599638254302	51	2024-04-23 15:43:03	--	None	⦸
☐	●	RyanlaSpeciale	2251799824568070	51	2024-04-23 15:38:59	--	None	⦸
☐	●	RyanlaSpeciale	6755399451934855	51	2024-04-23 15:31:35	--	None	⦸
☐	●	RyanlaSpeciale	4503599638241575	50	2024-04-23 15:12:34	--	None	⦸

2)User Acceptance Testing (UAT)

Objective: Validate the system's functionality and user-friendliness against the agreed acceptance criteria.

For This crucial Step I asked for Max a friend of mine that is an IT support Technician for an online game to test the system's functionality by himself and give me feedback about it;

Test ID	Test Description	Expected Outcome	Status
UAT-001	Test User-Friendliness	Intuitive and easy-to-use interface	Pass
UAT-002	Test Performance Benchmarks	System meets the defined performance benchmarks	Pass
UAT-003	Test System Completeness	All specified features are functional and implemented	Pass

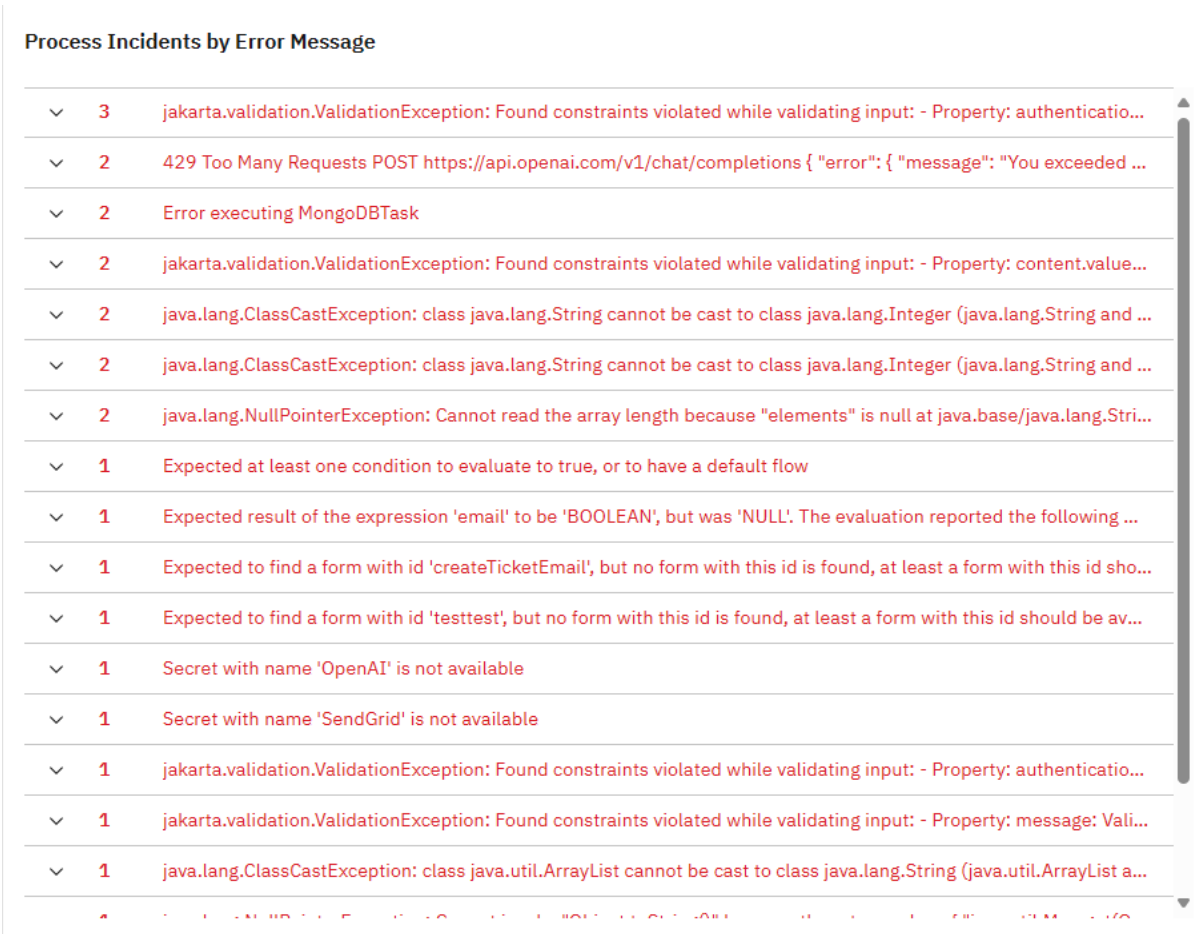
Risk Mitigation Document:

O

Objective: Identify potential risks, evaluate their impact and likelihood, and outline strategies to mitigate them throughout the development of the Ticketing System.

Risk ID	Risk Description	Likelihood	Impact	Mitigation Strategy	Status
R-001	Insufficient Camunda Knowledge	High	High	Extensive self-education, forums, and tutorials	Passed
R-002	Time Management	High	High	Detailed project plan and prioritization	Passed
R-003	Integration Challenges	Medium	High	Thorough testing and iterative development	Passed
R-004	Database Integration failure	Medium	Medium	Throughout testing and constant dev	Passed
R-005	SMTP Integration Failure	High	Medium	Robust error handling and testing	Passed
R-006	Performance Bottlenecks	Low	High	Performance testing and optimization	Passed

Post-Implementation Review:



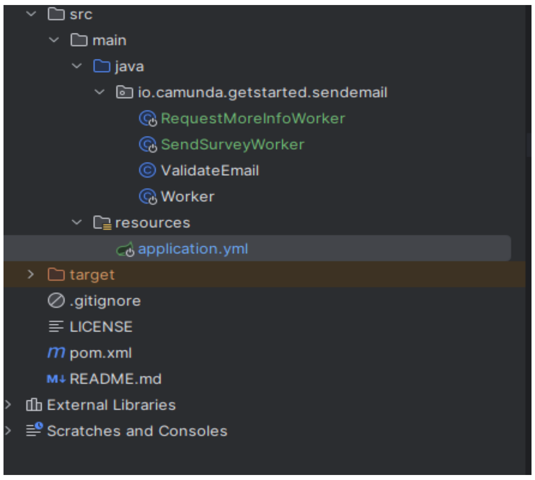
Process Incidents by Error Message

3	jakarta.validation.ValidationException: Found constraints violated while validating input: - Property: authenticatio...
2	429 Too Many Requests POST https://api.openai.com/v1/chat/completions { "error": { "message": "You exceeded ...
2	Error executing MongoDBTask
2	jakarta.validation.ValidationException: Found constraints violated while validating input: - Property: content.value...
2	java.lang.ClassCastException: class java.lang.String cannot be cast to class java.lang.Integer (java.lang.String and ...
2	java.lang.ClassCastException: class java.lang.String cannot be cast to class java.lang.Integer (java.lang.String and ...
2	java.lang.NullPointerException: Cannot read the array length because "elements" is null at java.base/java.lang.Stri...
1	Expected at least one condition to evaluate to true, or to have a default flow
1	Expected result of the expression 'email' to be 'BOOLEAN', but was 'NULL'. The evaluation reported the following ...
1	Expected to find a form with id 'createTicketEmail', but no form with this id is found, at least a form with this id sho...
1	Expected to find a form with id 'testtest', but no form with this id is found, at least a form with this id should be av...
1	Secret with name 'OpenAI' is not available
1	Secret with name 'SendGrid' is not available
1	jakarta.validation.ValidationException: Found constraints violated while validating input: - Property: authenticatio...
1	jakarta.validation.ValidationException: Found constraints violated while validating input: - Property: message: Vali...
1	java.lang.ClassCastException: class java.util.ArrayList cannot be cast to class java.lang.String (java.util.ArrayList a...

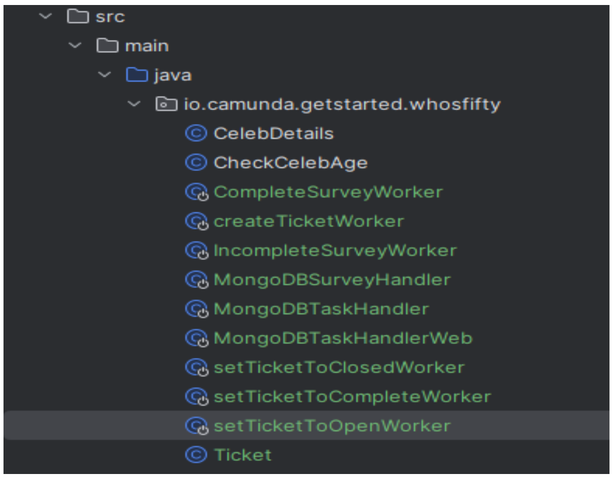
Version Management:

Git was employed for version control, facilitating tracking, documentation, and review of all changes before merging into the main branch. Despite the efficient version control system, managing branches and ensuring seamless integration posed challenges, particularly when multiple features were developed in parallel.

IntelliJIDEA: was employed for handling Job workers and Job handlers to empower the service tasks Required in the BPMN Model:

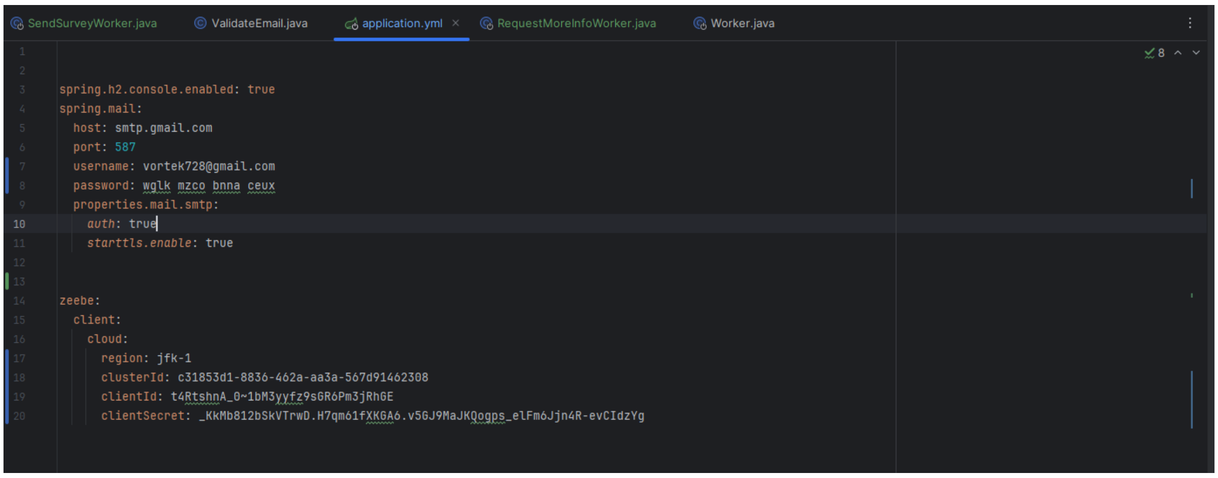


```
src
  main
    java
      io.camunda.getstarted.sendemail
        RequestMoreInfoWorker
        SendSurveyWorker
        ValidateEmail
        Worker
    resources
      application.yml
target
.gitignore
LICENSE
pom.xml
README.md
External Libraries
Scratches and Consoles
```



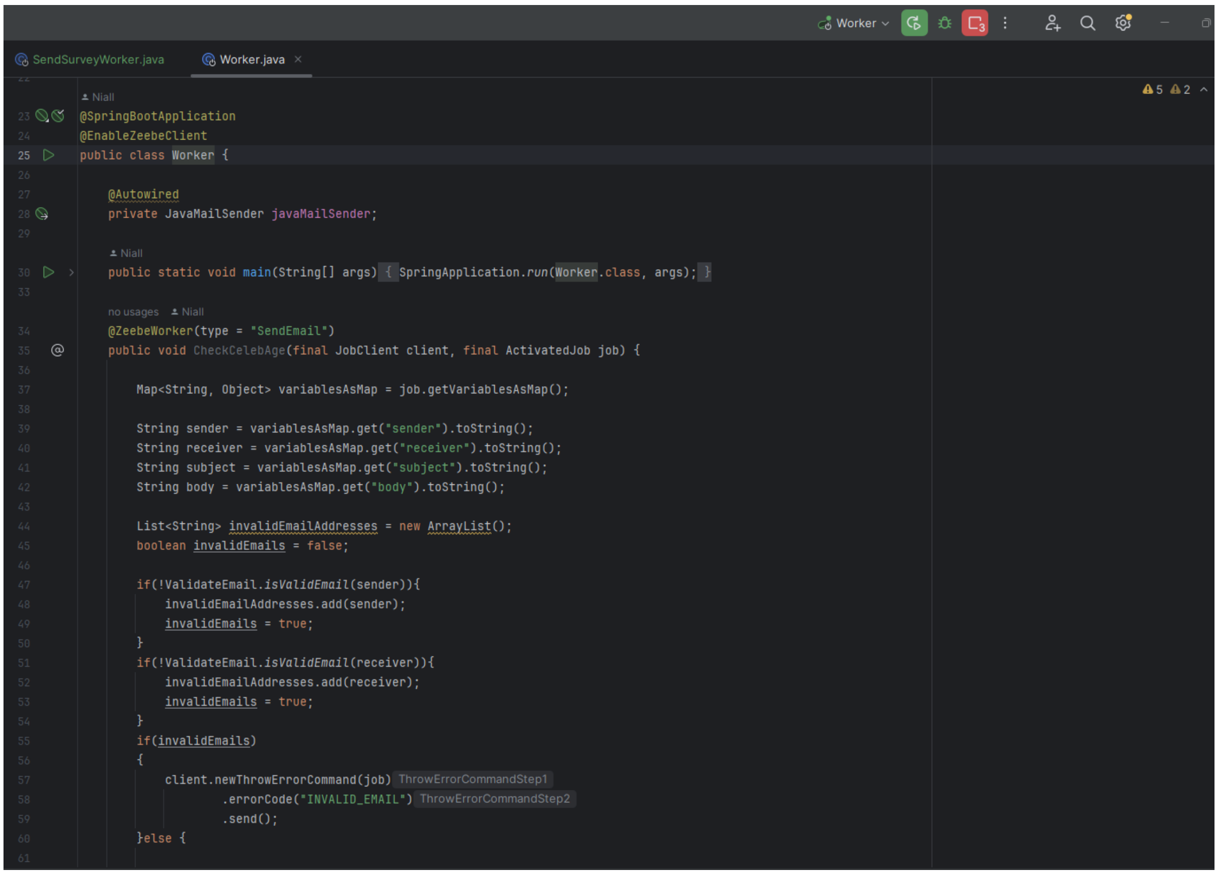
```
src
  main
    java
      io.camunda.getstarted.whosfifty
        CelebDetails
        CheckCelebAge
        CompleteSurveyWorker
        createTicketWorker
        IncompleteSurveyWorker
        MongoDBSurveyHandler
        MongoDBTaskHandler
        MongoDBTaskHandlerWeb
        setTicketToClosedWorker
        setTicketToCompleteWorker
        setTicketToOpenWorker
        Ticket
```

Application.yml config:

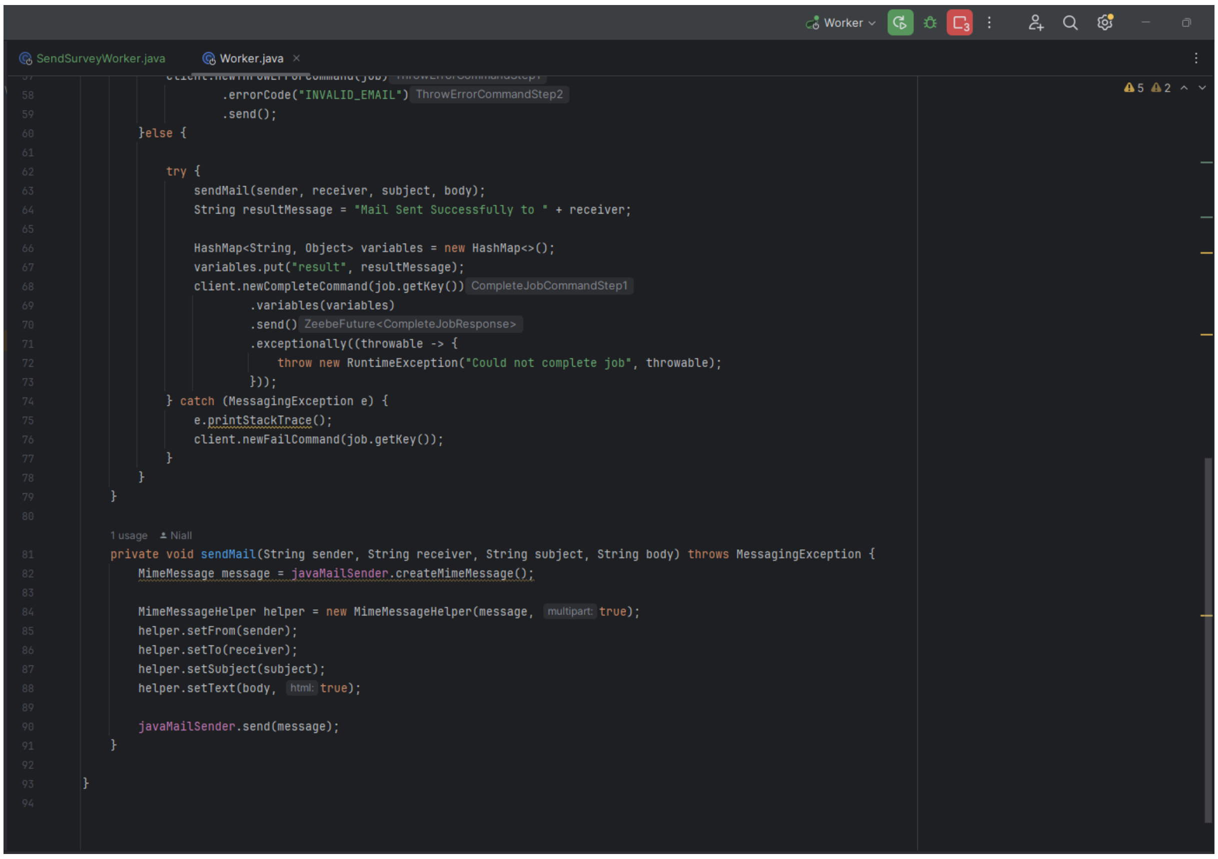


```yaml


spring.h2.console.enabled: true
spring.mail:
  host: smtp.gmail.com
  port: 587
  username: vortek728@gmail.com
  password: wglk mzco bnna ceux
  properties.mail.smtp:
    auth: true
    starttls.enable: true


zeebe:
  client:
    cloud:
      region: jfk-1
      clusterId: c31853d1-8836-462a-aa3a-567d91462308
      clientId: t4RtshnA_0~1bM3yyfz9sGR6Pm3jRhGE
      clientSecret: _KkMb812bSkVTrwD.H7qm61fXKGA6.v5GJ9MaJKQogps_elFm6Jjn4R-evCIdzYg
```

Email Worker:

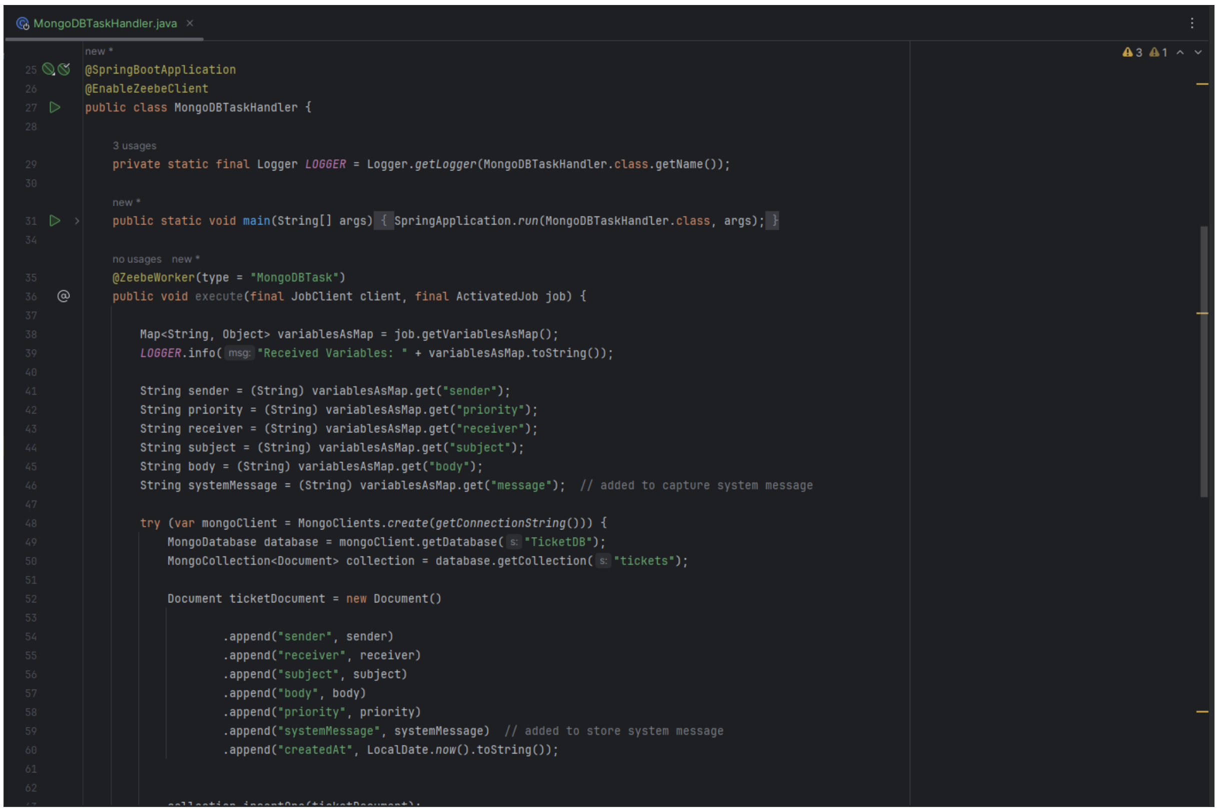


```java
@SpringBootApplication
@EnableZeebeClient
public class Worker {

    @Autowired
    private JavaMailSender javaMailSender;

    public static void main(String[] args) { SpringApplication.run(Worker.class, args); }

    @ZeebeWorker(type = "SendEmail")
    public void CheckCelebAge(final JobClient client, final ActivatedJob job) {

        Map<String, Object> variablesAsMap = job.getVariablesAsMap();

        String sender = variablesAsMap.get("sender").toString();
        String receiver = variablesAsMap.get("receiver").toString();
        String subject = variablesAsMap.get("subject").toString();
        String body = variablesAsMap.get("body").toString();

        List<String> invalidEmailAddresses = new ArrayList();
        boolean invalidEmails = false;

        if(!ValidateEmail.isValidEmail(sender)){
            invalidEmailAddresses.add(sender);
            invalidEmails = true;
        }
        if(!ValidateEmail.isValidEmail(receiver)){
            invalidEmailAddresses.add(receiver);
            invalidEmails = true;
        }
        if(invalidEmails)
        {
            client.newThrowErrorCommand(job)
                    .errorCode("INVALID_EMAIL")
                    .send();
        }else {

```

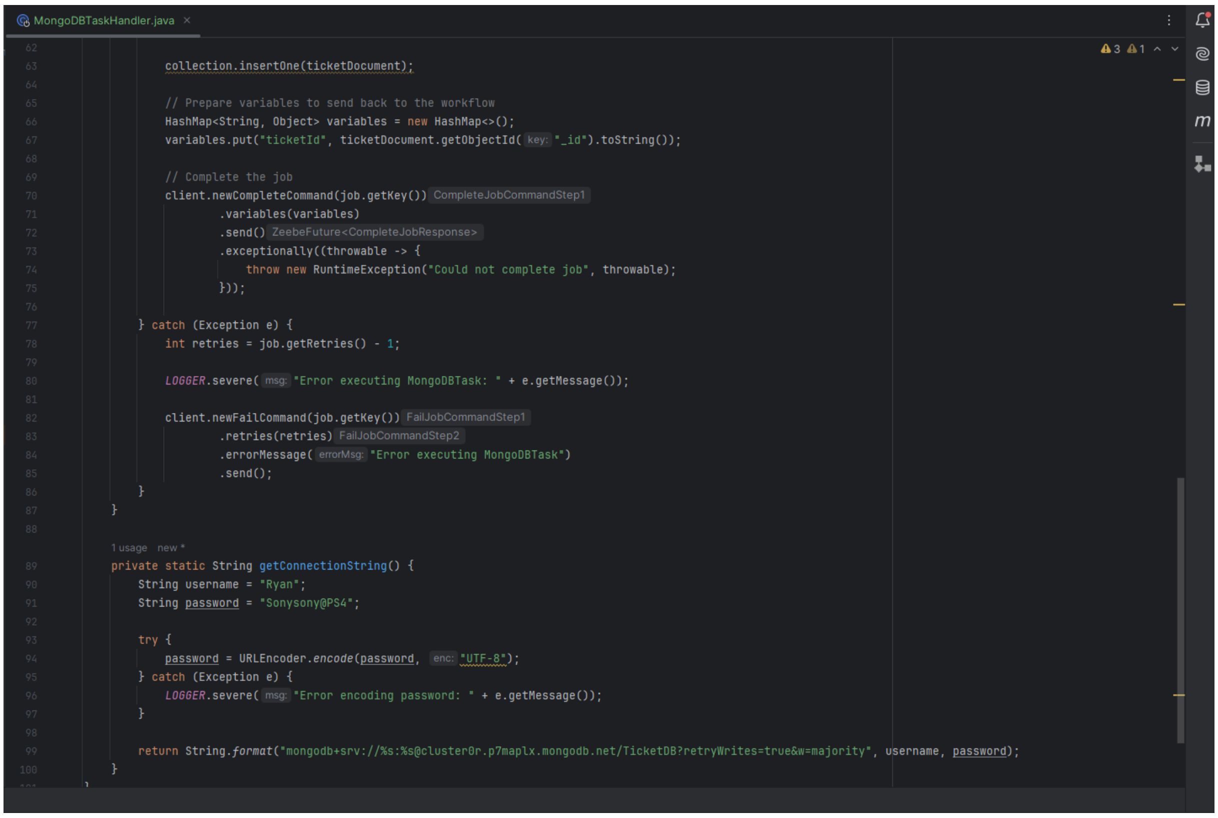


```java
                client.newThrowErrorCommand(job) ThrowErrorCommandStep1
                        .errorCode("INVALID_EMAIL") ThrowErrorCommandStep2
                        .send();
            }else {

                try {
                    sendMail(sender, receiver, subject, body);
                    String resultMessage = "Mail Sent Successfully to " + receiver;

                    HashMap<String, Object> variables = new HashMap<>();
                    variables.put("result", resultMessage);
                    client.newCompleteCommand(job.getKey()) CompleteJobCommandStep1
                            .variables(variables)
                            .send() ZeebeFuture<CompleteJobResponse>
                            .exceptionally((throwable -> {
                                throw new RuntimeException("Could not complete job", throwable);
                            }));
                } catch (MessagingException e) {
                    e.printStackTrace();
                    client.newFailCommand(job.getKey());
                }
            }
        }

        1 usage   Niall
        private void sendMail(String sender, String receiver, String subject, String body) throws MessagingException {
            MimeMessage message = javaMailSender.createMimeMessage();

            MimeMessageHelper helper = new MimeMessageHelper(message,  multipart: true);
            helper.setFrom(sender);
            helper.setTo(receiver);
            helper.setSubject(subject);
            helper.setText(body,  html: true);

            javaMailSender.send(message);
        }

    }
```

MongoDB Task Handle: The Mongo DB task Handles stores the User's Input when creating a ticket and stores the ticket with a unique ID.

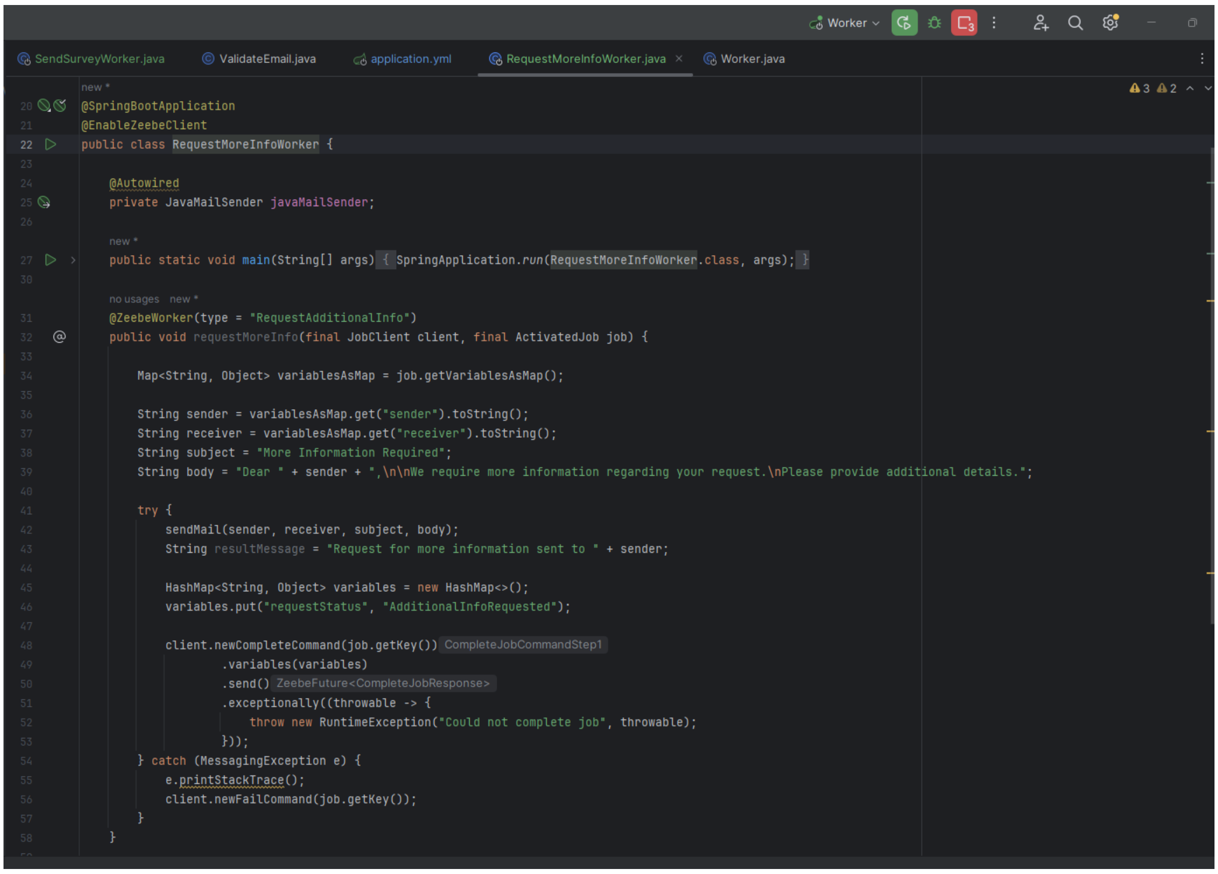


```java
new *
@SpringBootApplication
@EnableZeebeClient
public class MongoDBTaskHandler {

    3 usages
    private static final Logger LOGGER = Logger.getLogger(MongoDBTaskHandler.class.getName());

    new *
    public static void main(String[] args) { SpringApplication.run(MongoDBTaskHandler.class, args); }

    no usages   new *
    @ZeebeWorker(type = "MongoDBTask")
    public void execute(final JobClient client, final ActivatedJob job) {

        Map<String, Object> variablesAsMap = job.getVariablesAsMap();
        LOGGER.info( msg: "Received Variables: " + variablesAsMap.toString());

        String sender = (String) variablesAsMap.get("sender");
        String priority = (String) variablesAsMap.get("priority");
        String receiver = (String) variablesAsMap.get("receiver");
        String subject = (String) variablesAsMap.get("subject");
        String body = (String) variablesAsMap.get("body");
        String systemMessage = (String) variablesAsMap.get("message");  // added to capture system message

        try (var mongoClient = MongoClients.create(getConnectionString())) {
            MongoDatabase database = mongoClient.getDatabase( s: "TicketDB");
            MongoCollection<Document> collection = database.getCollection( s: "tickets");

            Document ticketDocument = new Document()

                    .append("sender", sender)
                    .append("receiver", receiver)
                    .append("subject", subject)
                    .append("body", body)
                    .append("priority", priority)
                    .append("systemMessage", systemMessage)  // added to store system message
                    .append("createdAt", LocalDate.now().toString());


            collection.insertOne(ticketDocument);
```



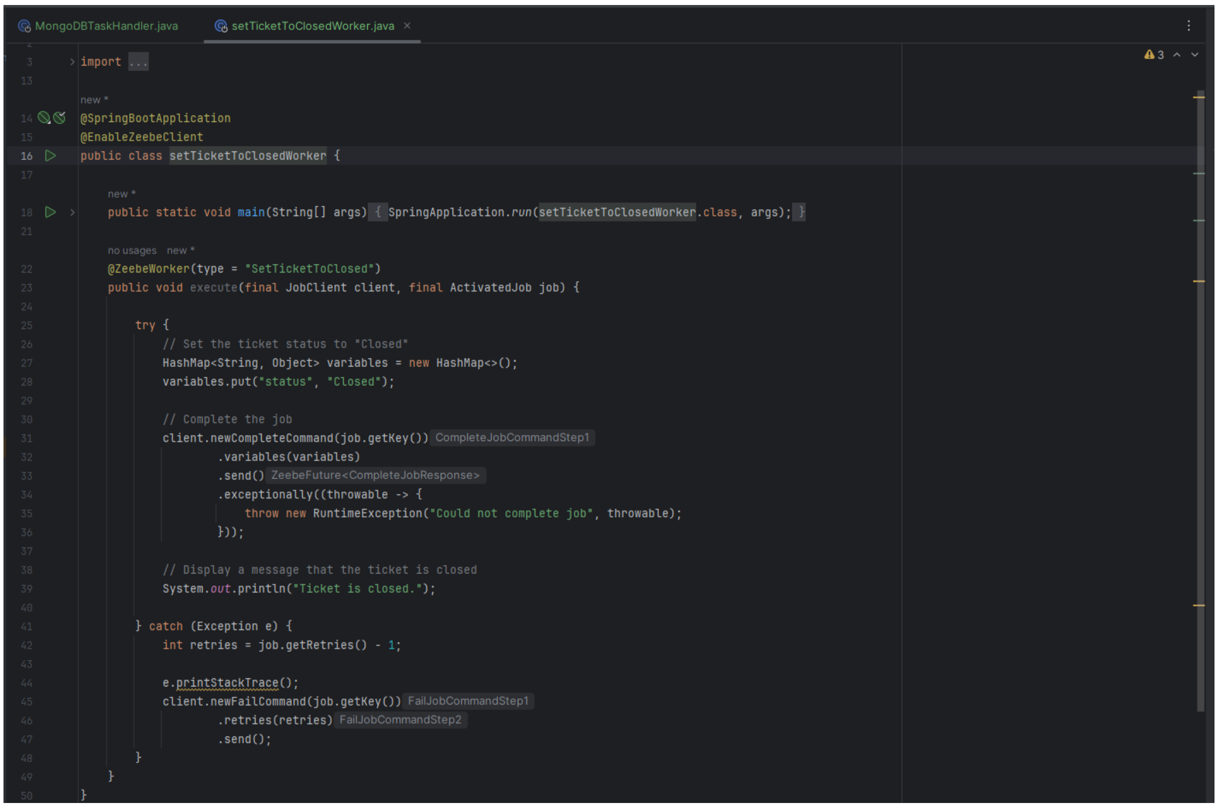
```java
            collection.insertOne(ticketDocument);

            // Prepare variables to send back to the workflow
            HashMap<String, Object> variables = new HashMap<>();
            variables.put("ticketId", ticketDocument.getObjectId("_id").toString());

            // Complete the job
            client.newCompleteCommand(job.getKey())
                    .variables(variables)
                    .send()
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));

        } catch (Exception e) {
            int retries = job.getRetries() - 1;

            LOGGER.severe("Error executing MongoDBTask: " + e.getMessage());

            client.newFailCommand(job.getKey())
                    .retries(retries)
                    .errorMessage("Error executing MongoDBTask")
                    .send();
        }
    }

    private static String getConnectionString() {
        String username = "Ryan";
        String password = "Sonysony@PS4";

        try {
            password = URLEncoder.encode(password, "UTF-8");
        } catch (Exception e) {
            LOGGER.severe("Error encoding password: " + e.getMessage());
        }

        return String.format("mongodb+srv://%s:%s@cluster0r.p7maplx.mongodb.net/TicketDB?retryWrites=true&w=majority", username, password);
    }
}
```

More Information Worker: Send An email with a notification letting know the user that he need to provide extra info



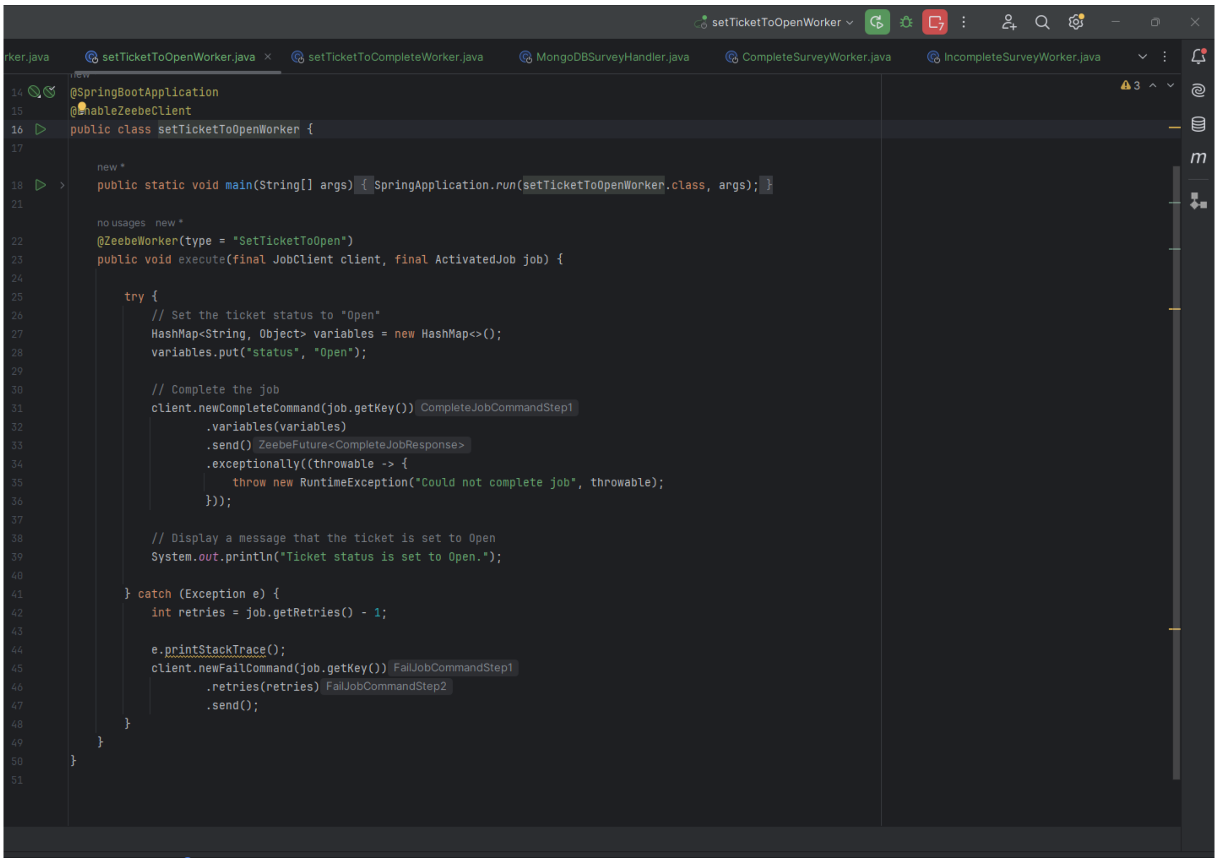
```java
@SpringBootApplication
@EnableZeebeClient
public class RequestMoreInfoWorker {

    @Autowired
    private JavaMailSender javaMailSender;

    public static void main(String[] args) { SpringApplication.run(RequestMoreInfoWorker.class, args); }

    @ZeebeWorker(type = "RequestAdditionalInfo")
    public void requestMoreInfo(final JobClient client, final ActivatedJob job) {

        Map<String, Object> variablesAsMap = job.getVariablesAsMap();

        String sender = variablesAsMap.get("sender").toString();
        String receiver = variablesAsMap.get("receiver").toString();
        String subject = "More Information Required";
        String body = "Dear " + sender + ",\n\nWe require more information regarding your request.\nPlease provide additional details.";

        try {
            sendMail(sender, receiver, subject, body);
            String resultMessage = "Request for more information sent to " + sender;

            HashMap<String, Object> variables = new HashMap<>();
            variables.put("requestStatus", "AdditionalInfoRequested");

            client.newCompleteCommand(job.getKey())
                    .variables(variables)
                    .send()
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));
        } catch (MessagingException e) {
            e.printStackTrace();
            client.newFailCommand(job.getKey());
        }
    }
```

Set Ticket to Closed: Sets the ticket status of resolution to closed



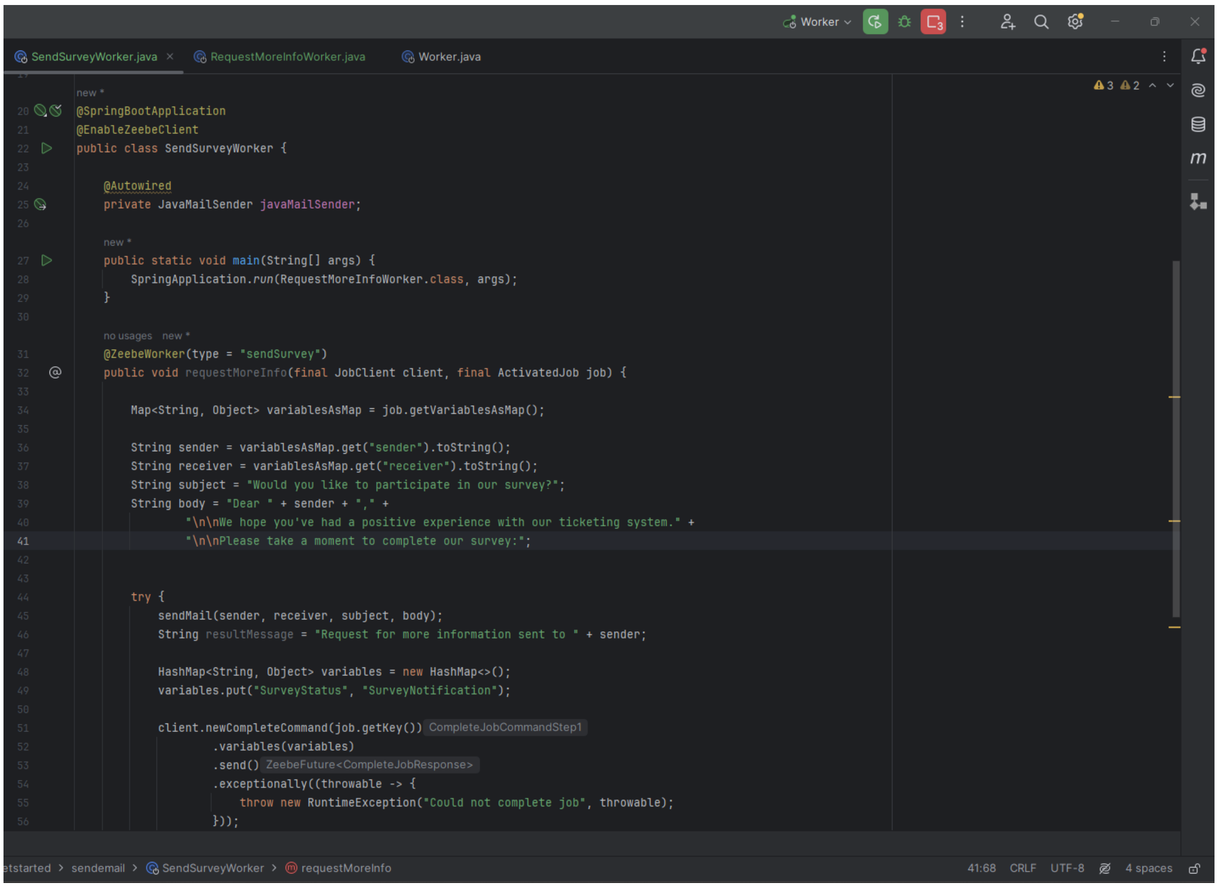
```java
import ...

@SpringBootApplication
@EnableZeebeClient
public class setTicketToClosedWorker {

    public static void main(String[] args) { SpringApplication.run(setTicketToClosedWorker.class, args); }

    @ZeebeWorker(type = "SetTicketToClosed")
    public void execute(final JobClient client, final ActivatedJob job) {

        try {
            // Set the ticket status to "Closed"
            HashMap<String, Object> variables = new HashMap<>();
            variables.put("status", "Closed");

            // Complete the job
            client.newCompleteCommand(job.getKey())
                    .variables(variables)
                    .send()
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));

            // Display a message that the ticket is closed
            System.out.println("Ticket is closed.");

        } catch (Exception e) {
            int retries = job.getRetries() - 1;

            e.printStackTrace();
            client.newFailCommand(job.getKey())
                    .retries(retries)
                    .send();
        }
    }
}
```

Set ticket back to Open:

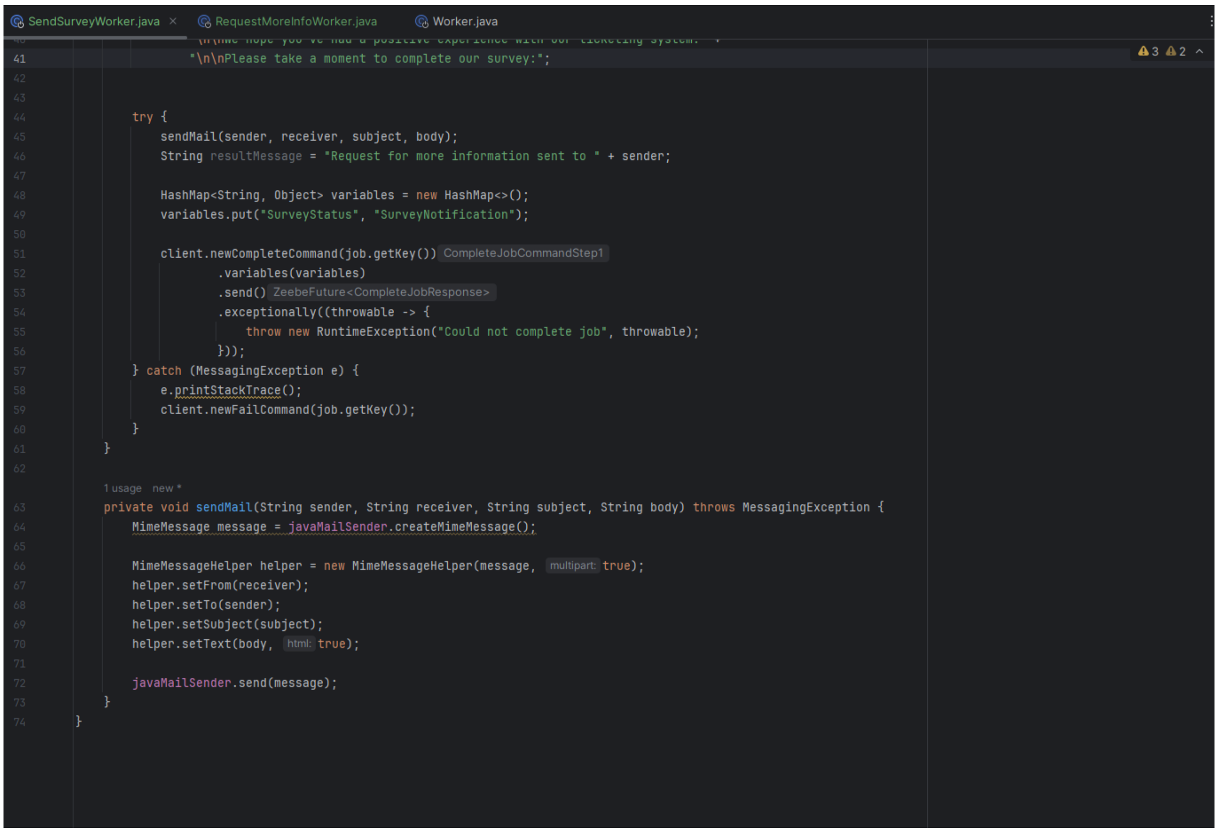

setTicketToOpenWorker
rker.java
setTicketToOpenWorker.java
setTicketToCompleteWorker.java
MongoDBSurveyHandler.java
CompleteSurveyWorker.java
IncompleteSurveyWorker.java

```java
@SpringBootApplication
@EnableZeebeClient
public class setTicketToOpenWorker {

    public static void main(String[] args) { SpringApplication.run(setTicketToOpenWorker.class, args); }

    @ZeebeWorker(type = "SetTicketToOpen")
    public void execute(final JobClient client, final ActivatedJob job) {

        try {
            // Set the ticket status to "Open"
            HashMap<String, Object> variables = new HashMap<>();
            variables.put("status", "Open");

            // Complete the job
            client.newCompleteCommand(job.getKey())
                    .variables(variables)
                    .send()
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));

            // Display a message that the ticket is set to Open
            System.out.println("Ticket status is set to Open.");

        } catch (Exception e) {
            int retries = job.getRetries() - 1;

            e.printStackTrace();
            client.newFailCommand(job.getKey())
                    .retries(retries)
                    .send();
        }
    }
}
```

Send Survey: Sends an email notification with the survey link to the survey


Worker
SendSurveyWorker.java
RequestMoreInfoWorker.java
Worker.java
⚠3 ⚠2
etstarted > sendemail > SendSurveyWorker > requestMoreInfo
41:68 CRLF UTF-8 4 spaces

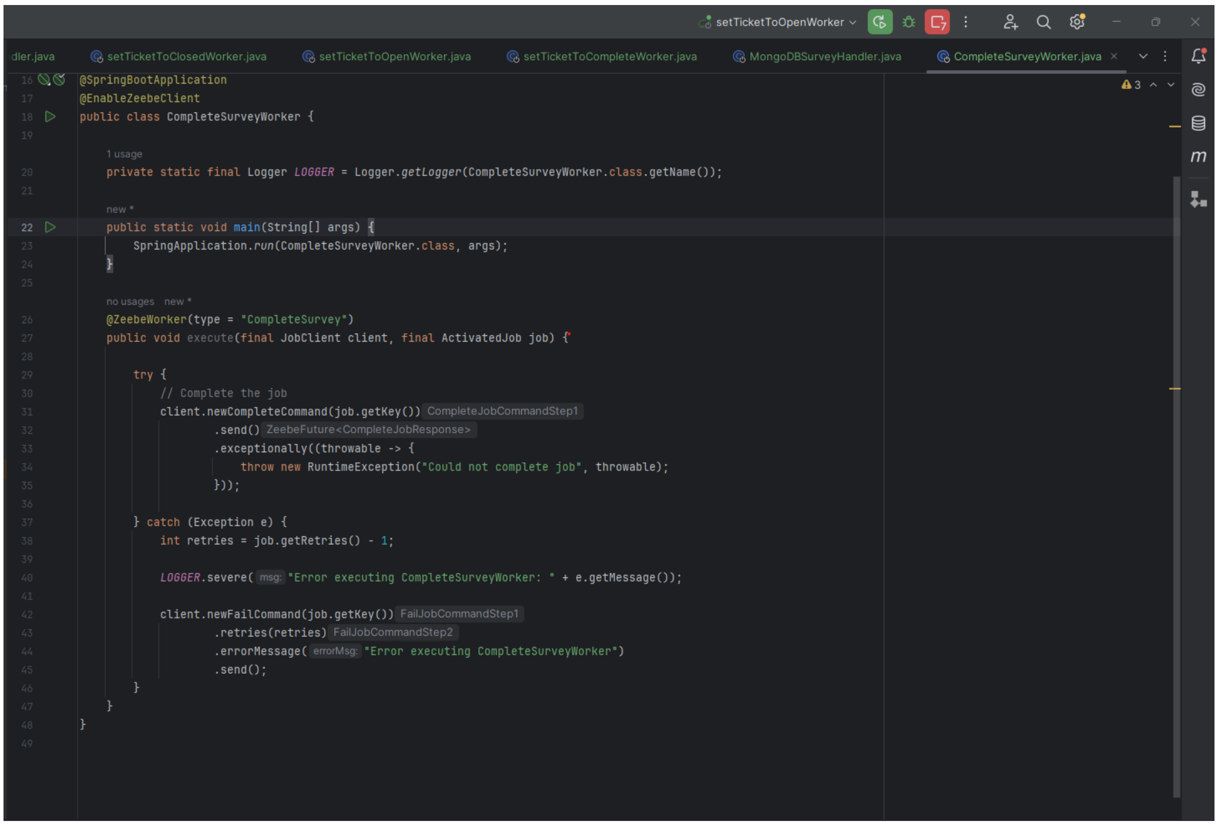
```java
new *
@SpringBootApplication
@EnableZeebeClient
public class SendSurveyWorker {

    @Autowired
    private JavaMailSender javaMailSender;

    new *
    public static void main(String[] args) {
        SpringApplication.run(RequestMoreInfoWorker.class, args);
    }

    no usages   new *
    @ZeebeWorker(type = "sendSurvey")
    public void requestMoreInfo(final JobClient client, final ActivatedJob job) {

        Map<String, Object> variablesAsMap = job.getVariablesAsMap();

        String sender = variablesAsMap.get("sender").toString();
        String receiver = variablesAsMap.get("receiver").toString();
        String subject = "Would you like to participate in our survey?";
        String body = "Dear " + sender + "," +
                "\n\nWe hope you've had a positive experience with our ticketing system." +
                "\n\nPlease take a moment to complete our survey:";


        try {
            sendMail(sender, receiver, subject, body);
            String resultMessage = "Request for more information sent to " + sender;

            HashMap<String, Object> variables = new HashMap<>();
            variables.put("SurveyStatus", "SurveyNotification");

            client.newCompleteCommand(job.getKey()) CompleteJobCommandStep1
                    .variables(variables)
                    .send() ZeebeFuture<CompleteJobResponse>
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));
```



```java
                "\n\nPlease take a moment to complete our survey:";


        try {
            sendMail(sender, receiver, subject, body);
            String resultMessage = "Request for more information sent to " + sender;

            HashMap<String, Object> variables = new HashMap<>();
            variables.put("SurveyStatus", "SurveyNotification");

            client.newCompleteCommand(job.getKey()) CompleteJobCommandStep1
                    .variables(variables)
                    .send() ZeebeFuture<CompleteJobResponse>
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));
        } catch (MessagingException e) {
            e.printStackTrace();
            client.newFailCommand(job.getKey());
        }
    }

    1 usage   new *
    private void sendMail(String sender, String receiver, String subject, String body) throws MessagingException {
        MimeMessage message = javaMailSender.createMimeMessage();

        MimeMessageHelper helper = new MimeMessageHelper(message,  multipart: true);
        helper.setFrom(receiver);
        helper.setTo(sender);
        helper.setSubject(subject);
        helper.setText(body,  html: true);

        javaMailSender.send(message);
    }
}
```

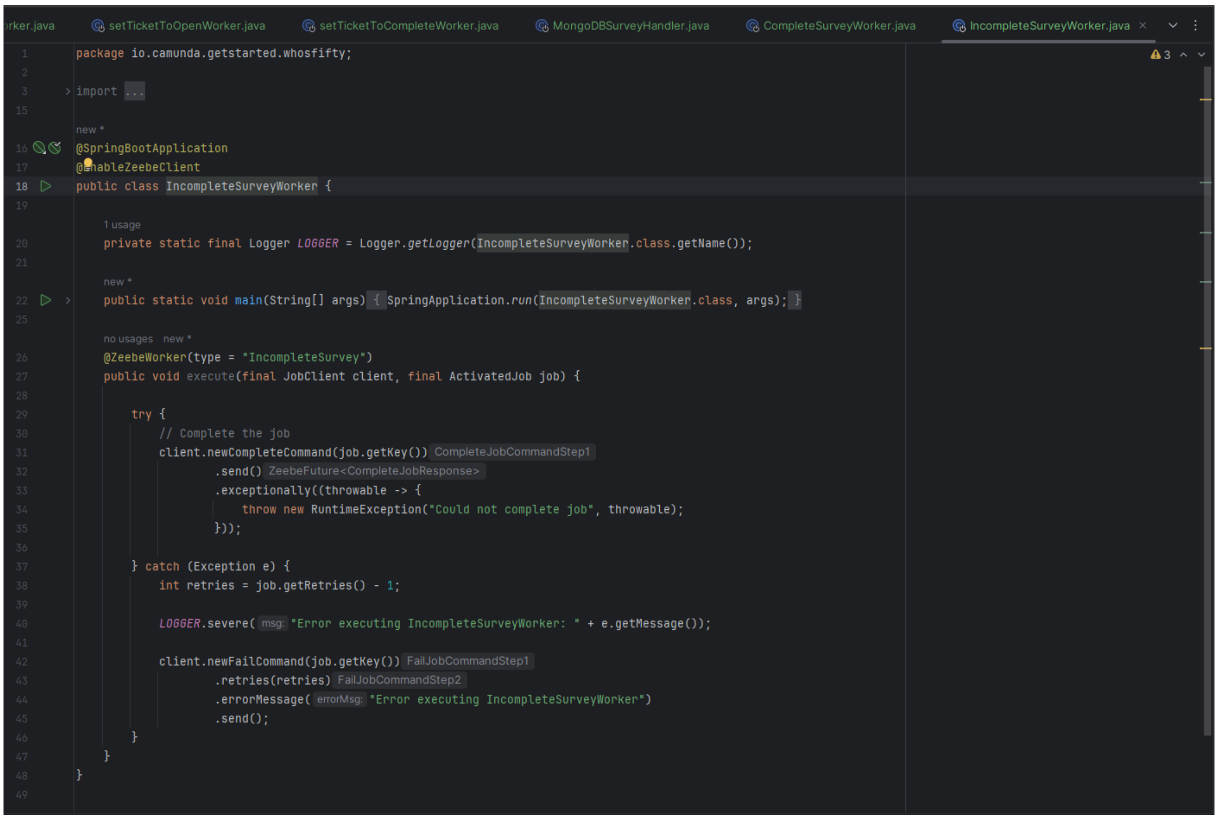
Set ticket back to Open:

Set Survey to Complete: Sets the survey status to complete after user opened the survey and filled the inputs:


setTicketToOpenWorker
dler.java
setTicketToClosedWorker.java
setTicketToOpenWorker.java
setTicketToCompleteWorker.java
MongoDBSurveyHandler.java
CompleteSurveyWorker.java

```java
@SpringBootApplication
@EnableZeebeClient
public class CompleteSurveyWorker {

    private static final Logger LOGGER = Logger.getLogger(CompleteSurveyWorker.class.getName());

    public static void main(String[] args) {
        SpringApplication.run(CompleteSurveyWorker.class, args);
    }

    @ZeebeWorker(type = "CompleteSurvey")
    public void execute(final JobClient client, final ActivatedJob job) {

        try {
            // Complete the job
            client.newCompleteCommand(job.getKey())
                    .send()
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));

        } catch (Exception e) {
            int retries = job.getRetries() - 1;

            LOGGER.severe("Error executing CompleteSurveyWorker: " + e.getMessage());

            client.newFailCommand(job.getKey())
                    .retries(retries)
                    .errorMessage("Error executing CompleteSurveyWorker")
                    .send();
        }
    }
}
```

Set Survey to Incomplete:


rker.java
setTicketToOpenWorker.java
setTicketToCompleteWorker.java
MongoDBSurveyHandler.java
CompleteSurveyWorker.java
IncompleteSurveyWorker.java

```java
package io.camunda.getstarted.whosfifty;

import ...

@SpringBootApplication
@EnableZeebeClient
public class IncompleteSurveyWorker {

    private static final Logger LOGGER = Logger.getLogger(IncompleteSurveyWorker.class.getName());

    public static void main(String[] args) { SpringApplication.run(IncompleteSurveyWorker.class, args); }

    @ZeebeWorker(type = "IncompleteSurvey")
    public void execute(final JobClient client, final ActivatedJob job) {

        try {
            // Complete the job
            client.newCompleteCommand(job.getKey())
                    .send()
                    .exceptionally((throwable -> {
                        throw new RuntimeException("Could not complete job", throwable);
                    }));

        } catch (Exception e) {
            int retries = job.getRetries() - 1;

            LOGGER.severe("Error executing IncompleteSurveyWorker: " + e.getMessage());

            client.newFailCommand(job.getKey())
                    .retries(retries)
                    .errorMessage("Error executing IncompleteSurveyWorker")
                    .send();
        }
    }
}
```

Project Requirements:

Priority Setting:

Each requirement was prioritized based on its importance to the overall functionality and user experience of the Ticketing System project:

-Functional Operational BPMN Model: Essential for the core functionality of the system, this requirement was of high priority.

-User could create tickets using the website: Crucial for user accessibility, this was also considered a high-priority requirement.

-User could create tickets using Email: Important for flexibility and user convenience, making it a high-priority feature.

-Email notifications: Critical for keeping users informed about ticket updates, rated as high priority.

-User's ticket is stored in a database: Fundamental for data persistence and retrieval, this was a high--priority requirement.

-Ticket Management: Vital for system organization and efficiency, making it a high-priority element.

-Forms to handle User's Input and user tasks: Necessary for collecting and processing user information, considered high priority.

-Job Handlers to handle service tasks: Essential for automating and managing backend processes, rated as high priority.

-Loops for error Handling: Important for system robustness and error recovery, making it a high-priority feature.

-Survey Creation: Important for gathering user feedback and improving services, considered a medium-priority requirement.

-Survey's data stored in a database: Required for data analysis and reporting, rated as medium priority.

-Process termination: Essential for ensuring the completion of system tasks, making it a high-priority element.

-Reporting and analytics of the overall project completion: Important for monitoring and improving system performance, considered a medium-priority requirement.

Acceptance Criteria:

The agreed acceptance criteria included:

-All specified features must be implemented and functional

-The system must be user-friendly and intuitive

-Instances has reach end event and complete the Process termination

-Performance benchmarks must be met.

Traceability Matrix: The following traceability matrix demonstrates how each requirement maps to the acceptance criteria and test cases:

Requirement	Acceptance Criteria	Test Case
Functional Operational BPMN Model	All specified features must be implemented and functional	TM-001, TM-002
User could create tickets using the website	System must be user-friendly and intuitive	TM-004
User could create tickets using Email	System must be user-friendly and intuitive	TM-006
Email notifications	Instances must reach end event and complete the Process termination	TM-007
User’s ticket is stored in a database	All specified features must be implemented and functional	TM-009
Ticket Management	All specified features must be implemented and functional	TM-010
Forms to handle User’s Input and user tasks	System must be user-friendly and intuitive	TM-013
Job Handlers to handle service tasks	All specified features must be implemented and functional	TM-014
Loops for error Handling	All specified features must be implemented and functional	TM-011
Survey Creation	System must be user-friendly and intuitive	TM-015
Survey’s data stored in a database	Performance benchmarks must be met	TM-016
Process termination	Instances must reach end event and complete the Process termination	TM-021
Reporting and analytics of the overall project completion	Performance benchmarks must be met	TM-002

Conclusion :

Throughout the development of the Ticketing System, a comprehensive and scalable solution was successfully designed, implemented, and deployed using Camunda BPM, Java, and various integrated technologies. The solo development methodology, combined with the extensive use of tools and technologies like Camunda Modeler, Docker Desktop, and MongoDB, enabled the creation of a robust and efficient ticketing system that meets the client's requirements.

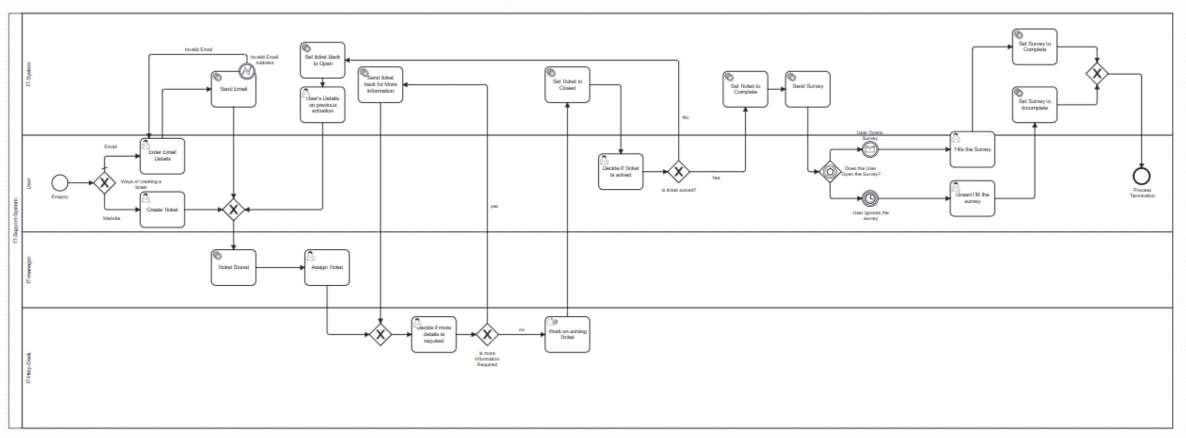
Despite the challenges faced, such as time management, technical complexities, and resource constraints, proactive risk mitigation, iterative development, and continuous testing ensured the timely delivery of a high-quality solution. The project's success can be attributed to the structured approach, comprehensive testing, and constant learning and adaptation throughout the development process.

Moving forward, continuous monitoring, regular updates, and incorporating user feedback will be crucial to maintaining the system's performance, scalability, and user satisfaction. The experience gained from this project has been invaluable, providing insights into effective project management, solo development methodologies, powering, creating Instances using Camunda, and the importance of leveraging the right tools and technologies to deliver innovative and efficient solutions.

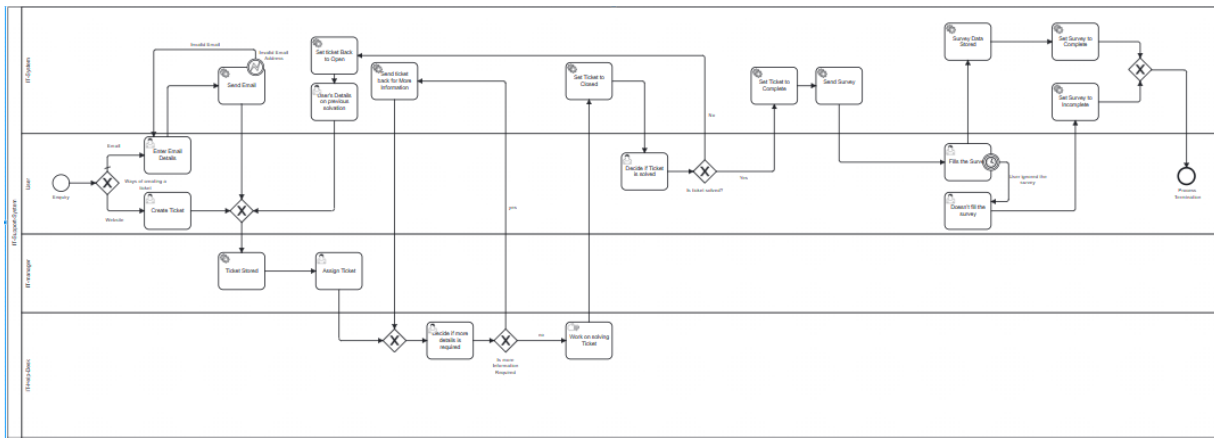
Overall, the Realised Ticketing System Project Showcases the capabilities and expertise in BPMN-based development but also emphasizes the commitment to delivering value-driven solutions that meet and exceed the client expectations Mr Dan.

Appendices:

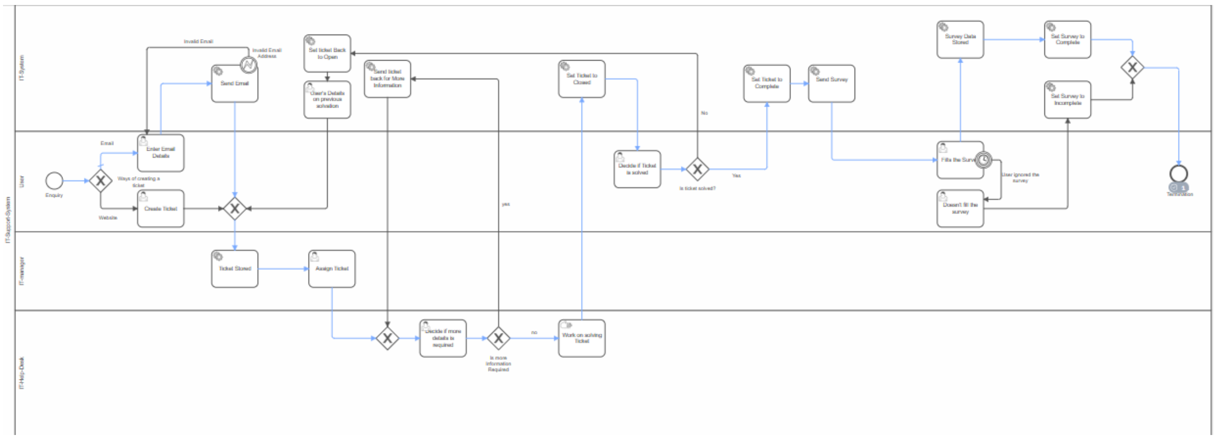
BPMN: Message CATCH Event for Survey Handling :



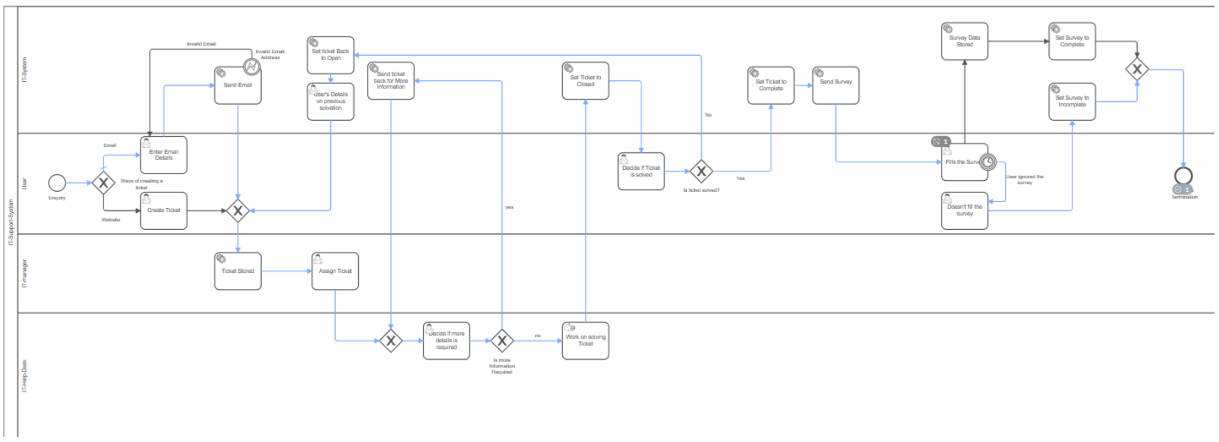
BPMN: *Timer Event for Survey Handling: (more optimal after testing)*



1st type of Process Termination:

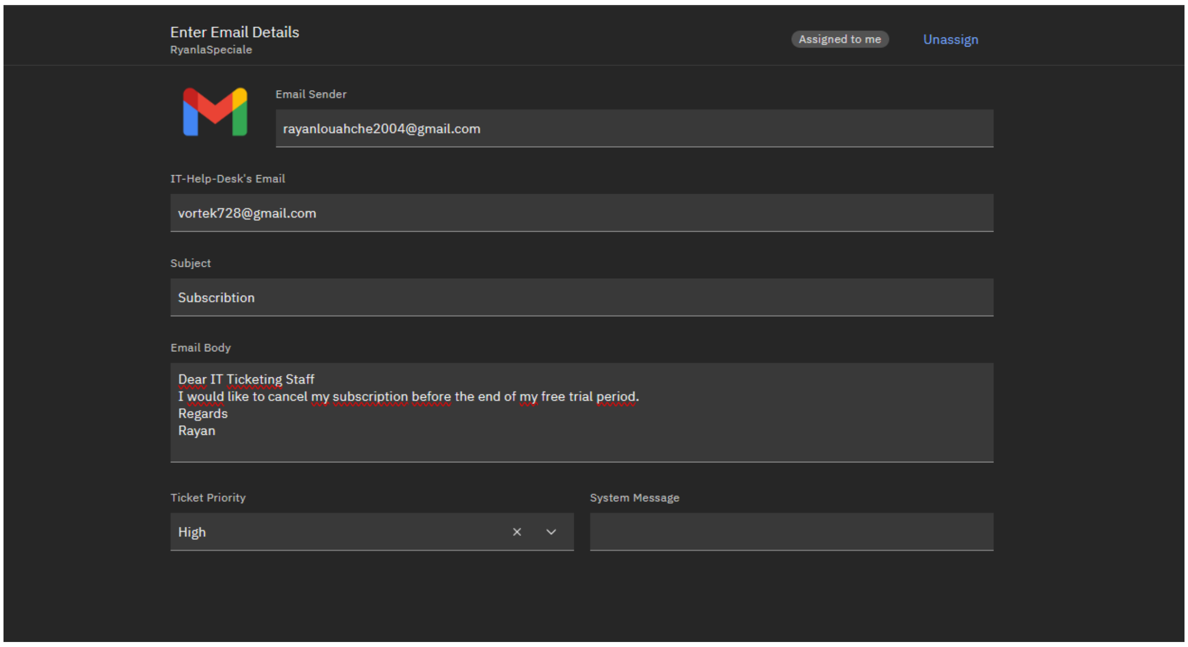


2nd type of Process Termination:



User Task Forms: *(Embedded and Linked):*

Create Email ticket, Enter Email Details:


Enter Email Details
RyanlaSpeciale

Email Sender: rayanlouahche2004@gmail.com

IT-Help-Desk's Email: vortek728@gmail.com

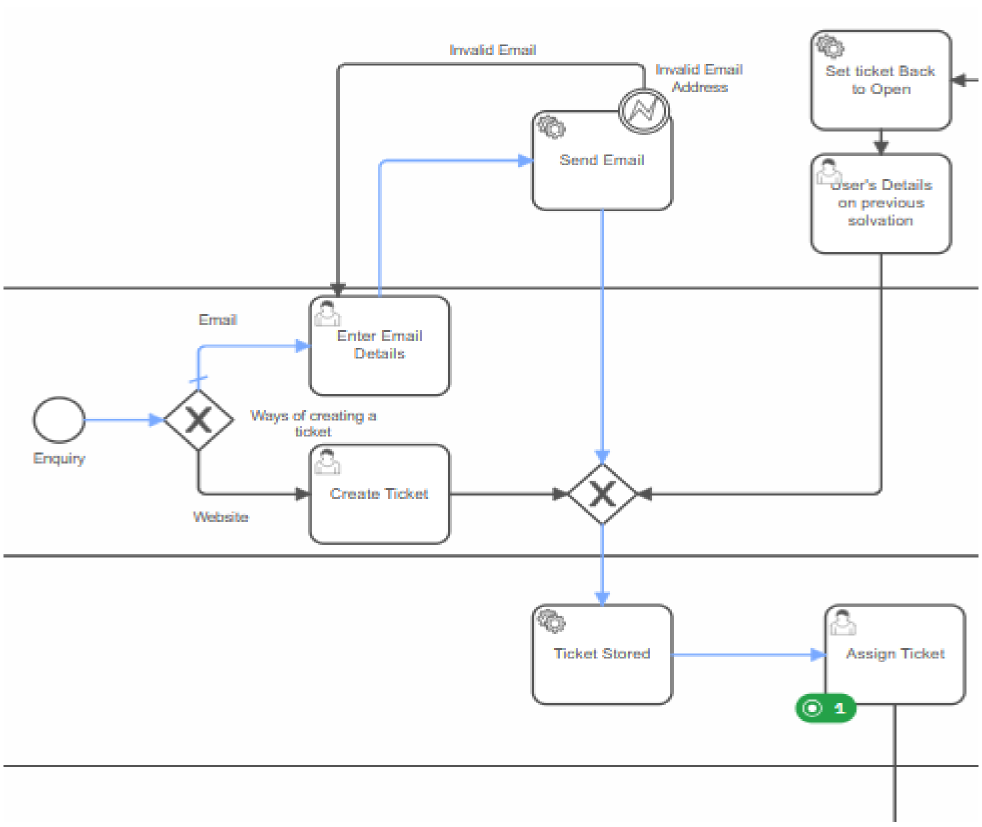
Subject: Subscribtion

Email Body:
Dear IT Ticketing Staff
I would like to cancel my subscription before the end of my free trial period.
Regards
Rayan

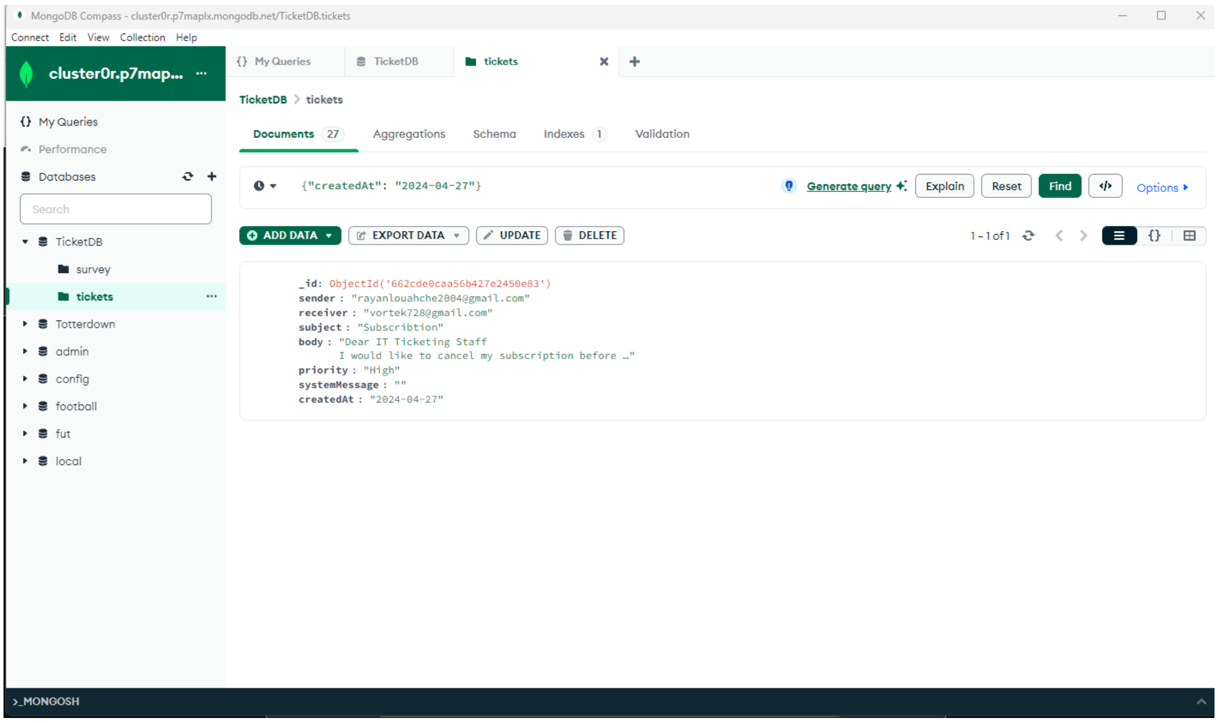
Ticket Priority: High

System Message:

Process Instance:



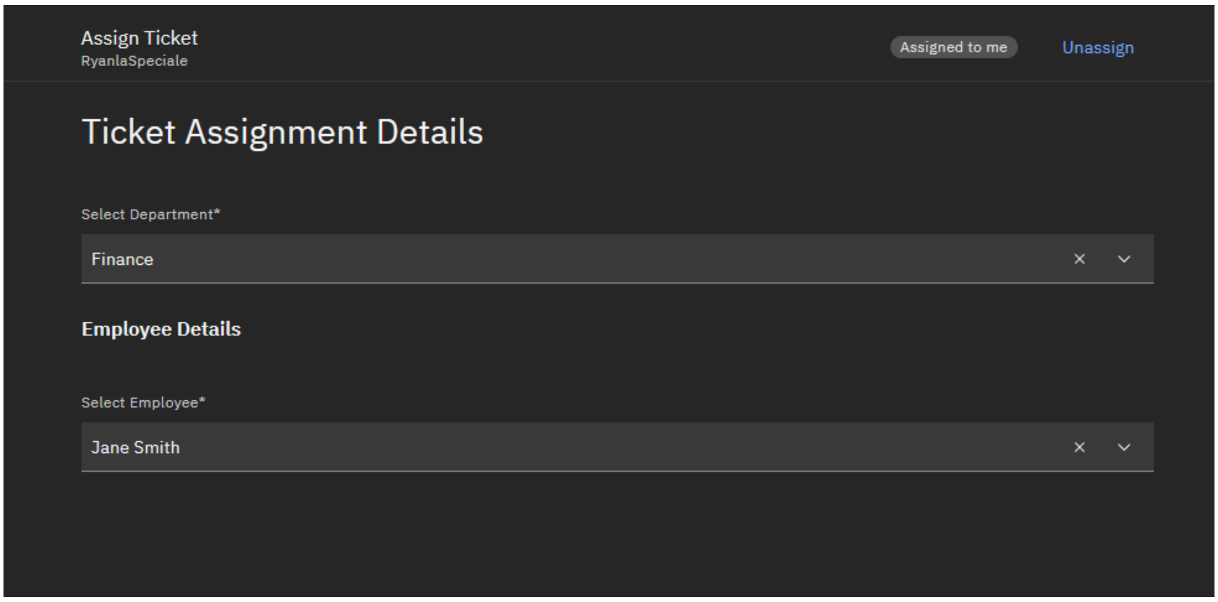
Ticket Stored:



```
{"createdAt": "2024-04-27"}
```

```
_id: ObjectId('662cde0caa56b427e2450e83')
sender : "rayanlouahche2004@gmail.com"
receiver : "vortek728@gmail.com"
subject : "Subscribtion"
body : "Dear IT Ticketing Staff
       I would like to cancel my subscription before …"
priority : "High"
systemMessage : ""
createdAt : "2024-04-27"
```

Assign Ticket:

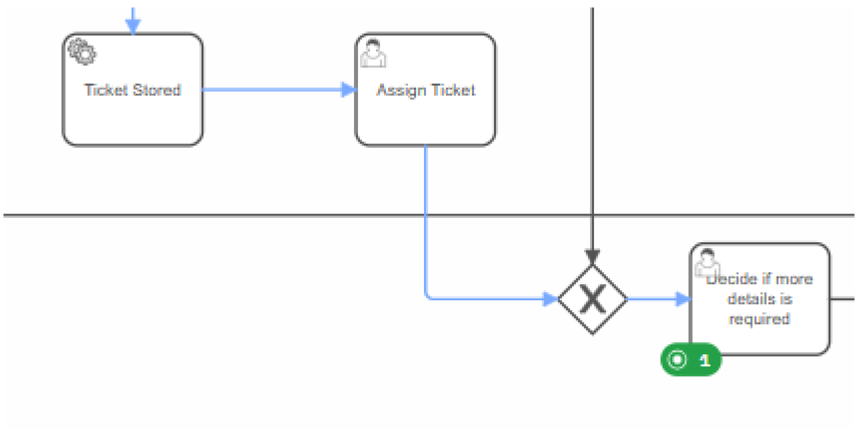

Assign Ticket
RyanlaSpeciale

Ticket Assignment Details

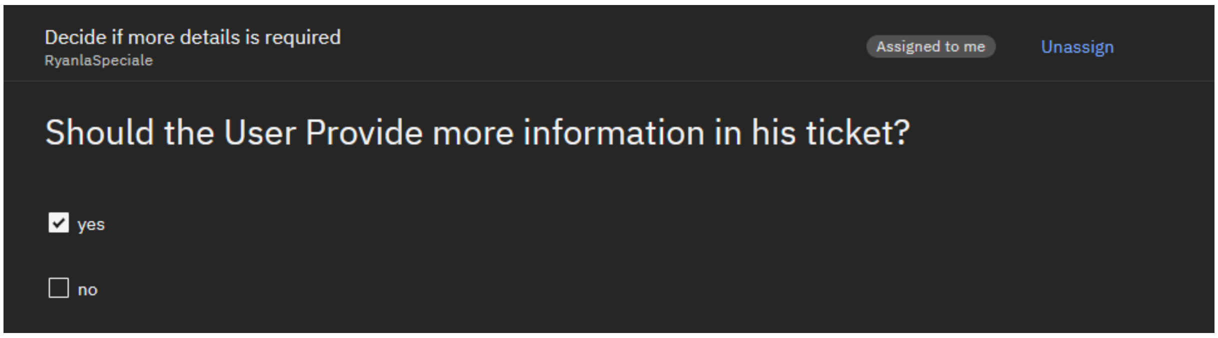
Select Department*
Finance

Employee Details

Select Employee*
Jane Smith



Should User provide more Information:

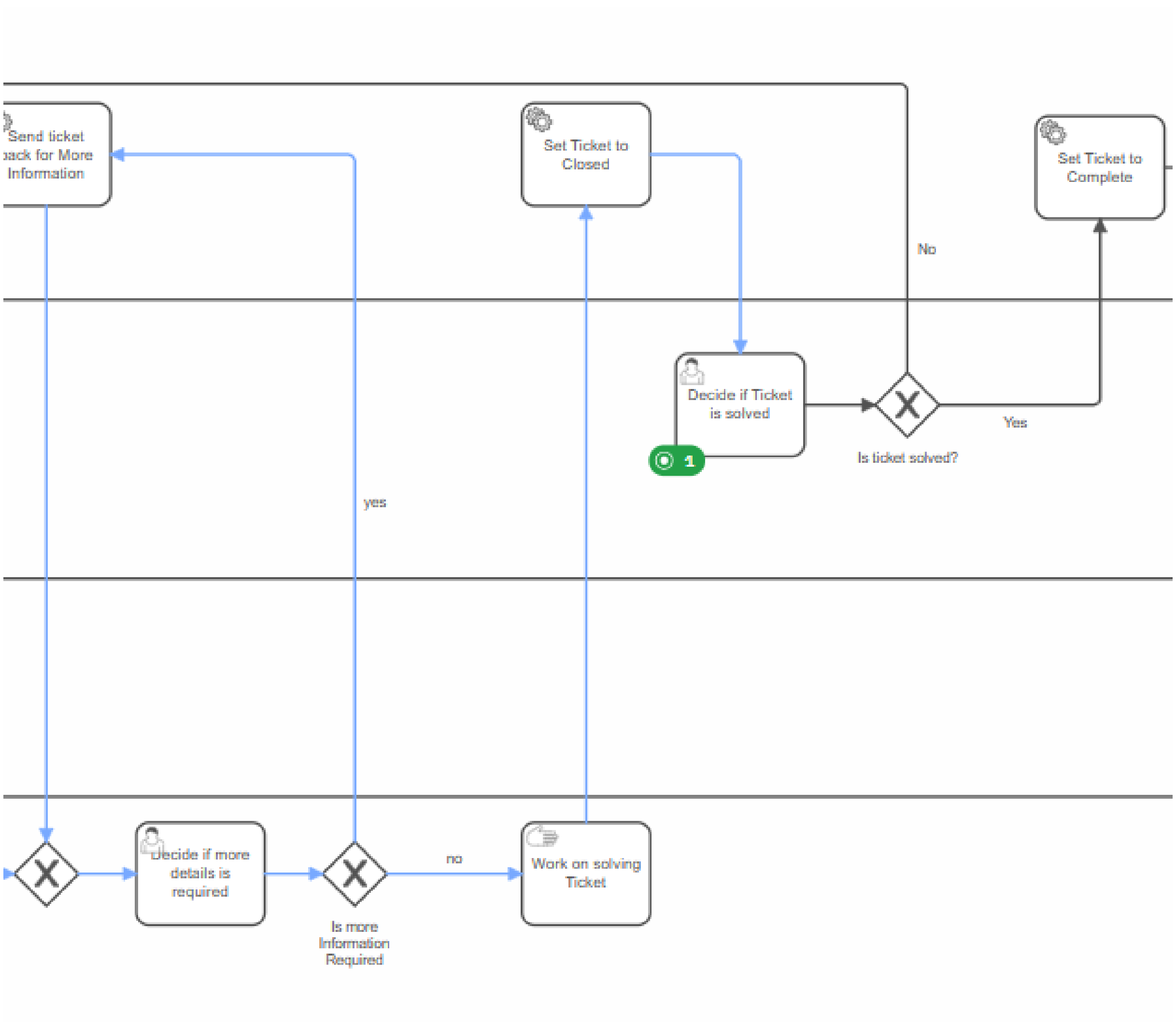

Decide if more details is required
RyanlaSpeciale

Should the User Provide more information in his ticket?

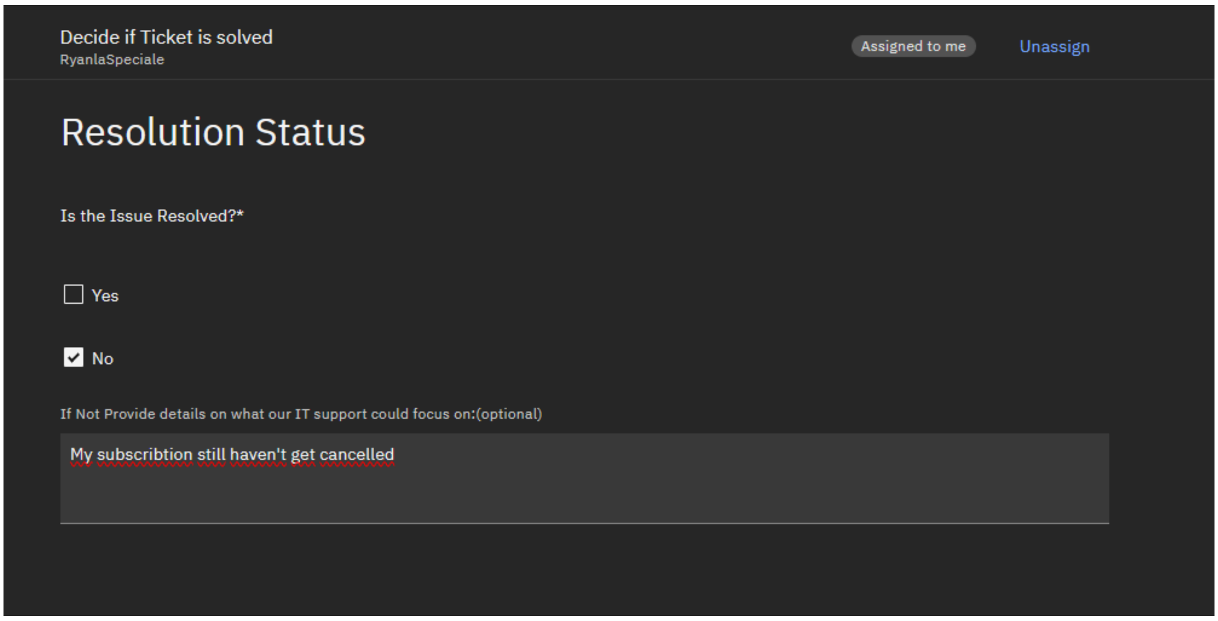
☑ yes

☐ no

Process Instance:


Send ticket back for More Information
Set Ticket to Closed
Set Ticket to Complete
No
Decide if Ticket is solved
1
Is ticket solved?
Yes
yes
Decide if more details is required
no
Work on solving Ticket
Is more Information Required

Decide if ticket is solved:



Decide if Ticket is solved
RyanlaSpeciale

Assigned to me Unassign

Resolution Status

Is the Issue Resolved?*

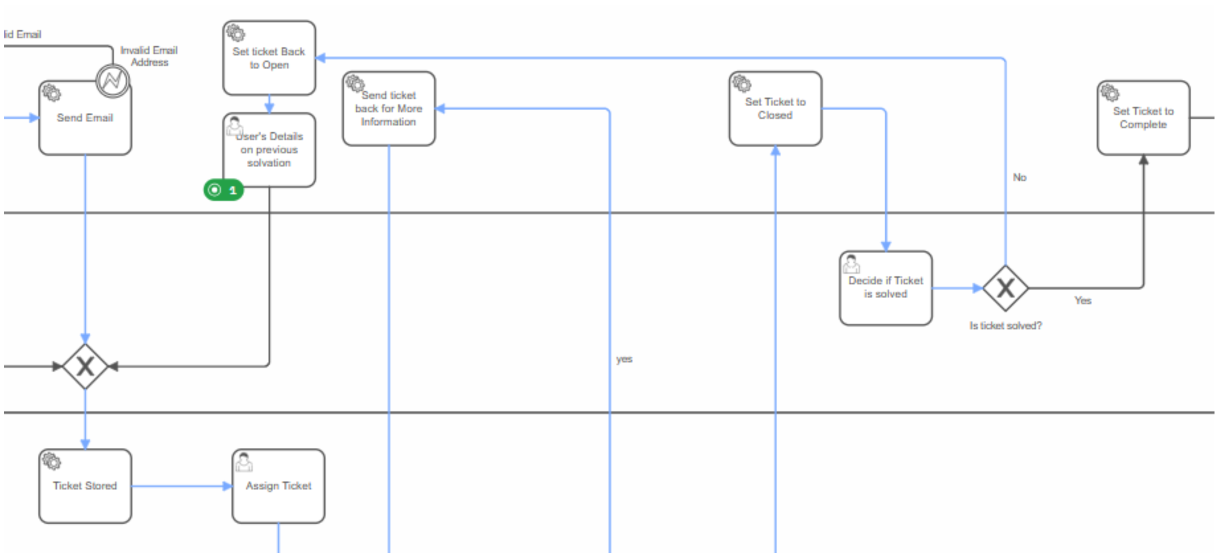
☐ Yes

☑ No

If Not Provide details on what our IT support could focus on:(optional)

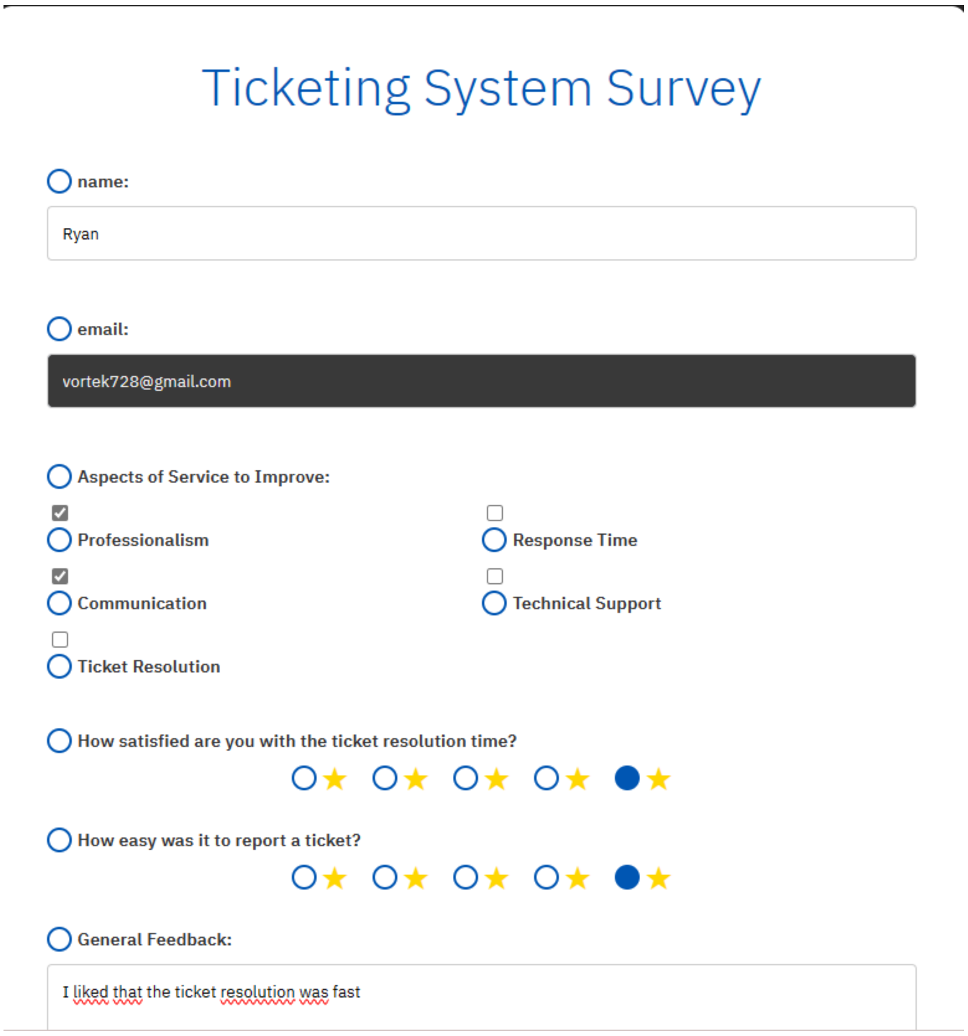
My subscribtion still haven't get cancelled

Process Instance:



IT checks User's details after refuse of resolution:

User Filled the survey:



Ticketing System Survey

name:

Ryan

email:

vortek728@gmail.com

Aspects of Service to Improve:

- [✓ Professionalism]
- Response Time
- [✓ Communication]
- Technical Support
- Ticket Resolution

How satisfied are you with the ticket resolution time?

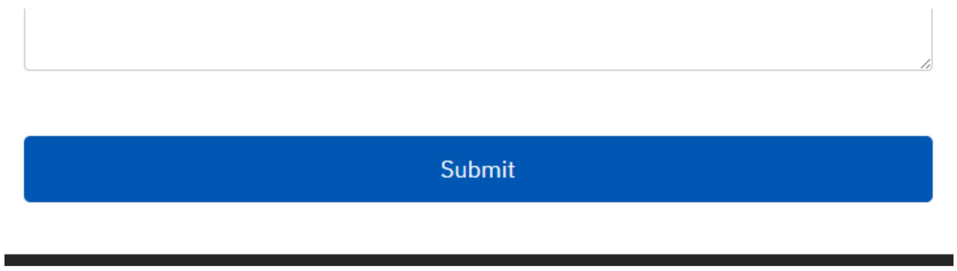
○★ ○★ ○★ ○★ ●★

How easy was it to report a ticket?

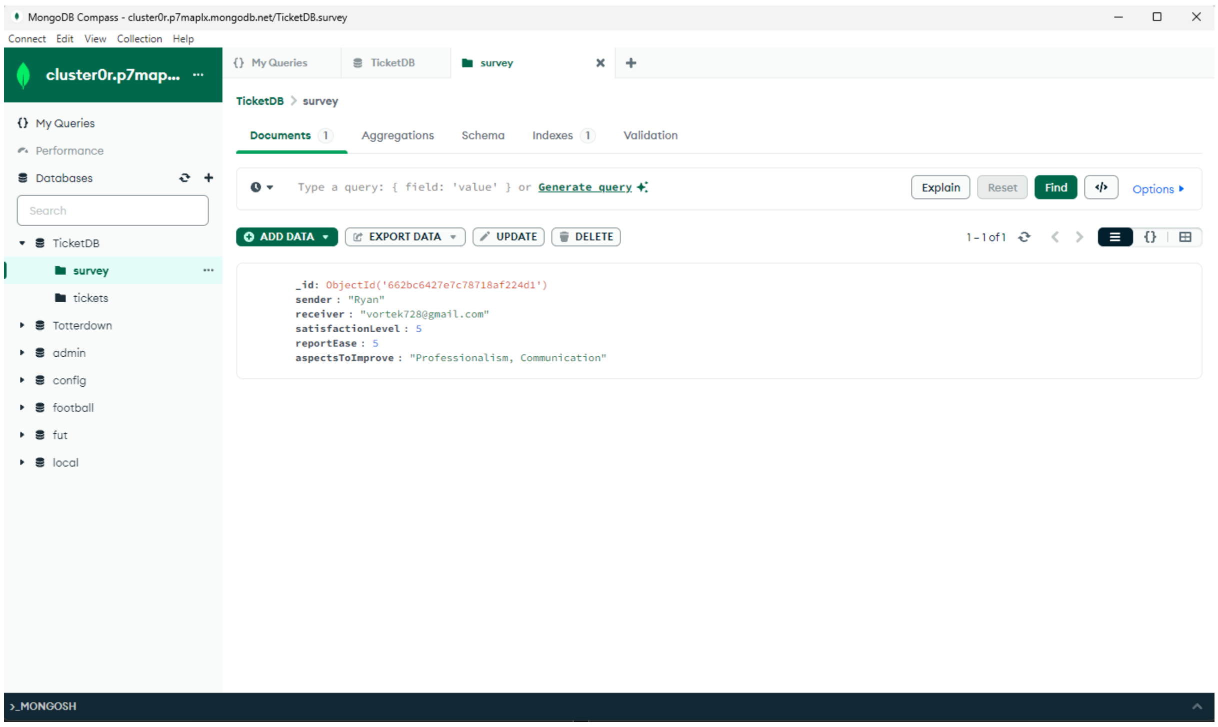
○★ ○★ ○★ ○★ ●★

General Feedback:

I liked that the ticket resolution was fast

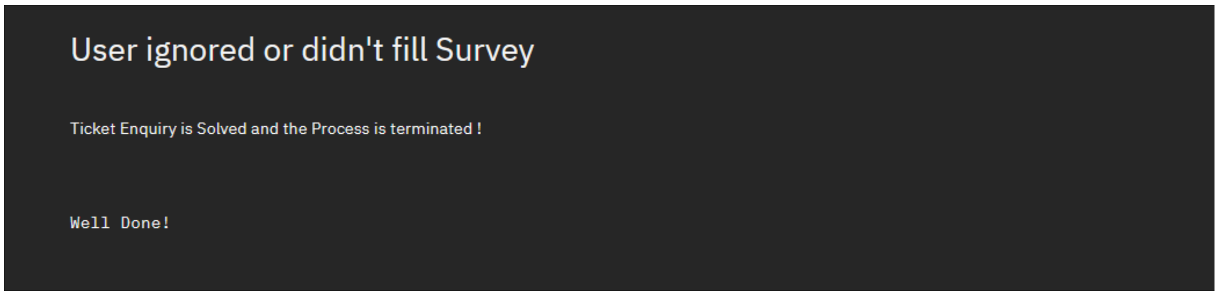


Survey Data Stored:



```
_id: ObjectId('662bc6427e7c78718af224d1')
sender : "Ryan"
receiver : "vortek728@gmail.com"
satisfactionLevel : 5
reportEase : 5
aspectsToImprove : "Professionalism, Communication"
```

User Ignored the Survey:

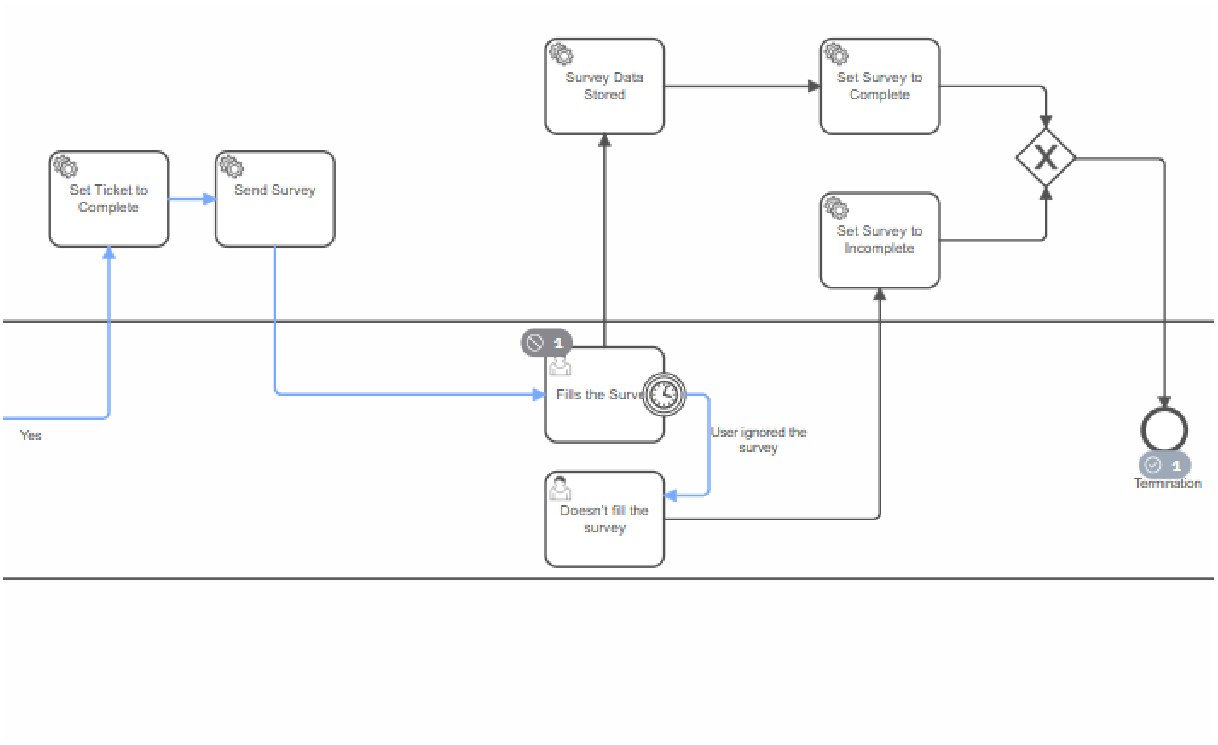


User ignored or didn't fill Survey

Ticket Enquiry is Solved and the Process is terminated !

```
Well Done!
```

Process Instance: *(Process Termination):*



Referencing:

Camunda:

Camunda. (n.d.). In Wikipedia. Retrieved January 15, 2024, from https://en.wikipedia.org/wiki/Camunda

BPMN (Business Process Model and Notation):

Object Management Group. (2011). Business Process Model and Notation (BPMN), Version 2.0. Available at: https://www.omg.org/spec/BPMN/2.0/

Service Task:

Camunda. (n.d.). Service Task. Retrieved January 15, 2024, from https://docs.camunda.org/manual/7.16/reference/bpmn20/tasks/service-task/

User Task:

Camunda. (n.d.). User Task. Retrieved January 15, 2024, from https://docs.camunda.org/manual/7.16/reference/bpmn20/tasks/user-task/

Process Assessment:

- Kirchmer, M., & Franz, R. (2009). The Process Assessment Model (PAM). Springer Berlin Heidelberg.

Continuous Assessment:

Duffy, T. M., & Kirkley, J. R. (2004). Learner-centered theory and practice in distance education: Cases from higher education. Lawrence Erlbaum Associates Publishers.

Process Modeling and Analysis:

- Reijers, H. A., & Liman Mansar, S. (2005). Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega, 33(4), 283-306.

Workflow Management Systems:

- van der Aalst, W. M., & van Hee, K. M. (2002). Workflow management: Models, methods, and systems. MIT press.

Agile Methodology:

- Highsmith, J. (2002). Agile software development ecosystems. Addison-Wesley.

Risk Management:

- Hillson, D., & Murray-Webster, R. (2017). Understanding and managing risk attitude. Routledge.

Project Management:

- Kerzner, H. (2017). Project management best practices: Achieving global excellence. John Wiley & Sons.

8. Software Development:

    - Sommerville, I. (2015). Software engineering. Pearson Education Limited.