Workflow Engine

Jirapa Lapanant
Pimpisa Watthanavarangkul
Chusanart Phongsuwan

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2016
KMITL-2017-IC-B-003-007

Thesis - Academic Year 2016

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: Workflow Engine

Authors:

1. Miss Jirapa Lapanant Student ID: 56090007
2. Miss Pimpisa Watthanavarangkul Student ID: 56090019
3. Miss Chusanart Phongsuwan Student ID: 56090039

Approved for Submission

V. Hirankitti

(Asst. Prof. Dr.Visit Hirankitti)
Advisor

Date 4/6/2017

Abstract

The world is moving towards Industry 4.0 where production process in a factory is fully automated by deploying robots. Analogously, in the world of web technology we could innovate a new way of automating a business process by an automatic workflow. This can be done by what we call "Workflow Engine".

The aim of this project is to automate business workflows on the web which results in improving people collaboration and increasing productivity. Workflow engine allows people to program their business activities into BPMN diagrams and let them connect web services together with provided simple input and output forms using the diagram as the programming tool. The application also provides user authentication in order to prevent user's private information and files.

The client side of the workflow engine is developed using Django Web Framework and Django Channel.

Acknowledgements

The best feeling in life, is knowing that you are not alone. That you have someone to rely on when things get rough. That there are some others willing to lend you a hand when facing troubles. So in this page of the thesis, we would like to thanks each others for holding on and for the discussions and every decision making that we have been through. Although some may be regrettable but in the end, we managed to unite and pull through any obstacle that has been thrown our way.

This thesis can also never been complete without our advisor, Dr.Visit Hirankitti. It had been a privilege to work with him and to have the opportunity to listened to all of his advises. We really appreciated that a hard-working man as himself still make some times for us in order to help and uplifted the efficiency of this project.

Furthermore, we would like to say thank you to all those people at International College, King Mongkut's Institute of Technology Ladkrabang. All college staffs who were always there when we had questions. All the professors who were always ready to support and help in every possible ways. Including all the friends and classmates for all the suggestions and interests. The people behind this thesis are more than just us, but it is everyone.

Lastly, we would like to thanks our parents and all other members of the family for supporting us spiritually throughout studying all 4 years, developing the project, writing this thesis, and our lives in general.

Table of Contents

Page

Abstract ..... **I**

Acknowledgement ..... **II**

Table of Contents ..... **III**

List of Tables ..... **IV**

List of Figures ..... **VI**

Chapter 1: Introduction **1**

1.1) Motivations 1

1.2) Objectives 1

1.3) Problem Description 2

1.4) Scope of Works 2

1.5) Contributions 2

1.6) Thesis Structure 3

Chapter 2: Related Works **5**

2.1) Microsoft Flow 5

2.2) If This Then That (IFTTT) 6

2.3) Camunda BPM 6

2.4) Alfresco 7

2.5) Comparing Table 8

Chapter 3: Business Workflow **9**

3.1) Modelling of Business Workflow 9

3.2) Business Process Model and Notation (BPMN) 9

3.2.1) BPMN Notations 9

3.2.2) Execution/ Interpretation 17

3.2.3) Workflow Automation 17

Chapter 4: Software Development 18

4.1) Requirements 18

4.1.1) Functional Requirements 18

4.1.2) Non - Functional Requirements 19

4.1.3) Use Case 20

4.1.4) Sequence Diagram 30

4.2) Designs 34

4.2.1) System Architectures 34

4.2.2) Class Diagram for User Model 35

4.2.3) Class Diagram for the Graph Representation 36

4.2.4) Workflow Database Diagram 37

Chapter 5: Web Application Development 38

5.1) Django Web Framework 38

5.2) Django Channel 39

5.3) WebSocket 40

5.4) Software Development Tools Used in This Project.....42

Chapter 6: Workflow Engine 44

6.1) The Tools of Drawing BPMN Diagrams 44

6.1.1) Form Creation 46

6.2) Converting BPMN XML to its Graph Representation 46

6.3) Execution of Graph-based Workflows 51

6.3.1) Decision Making.....54

6.3.2) Joining and Splitting of Parallel Gateway.....55

6.3.3) Event Handling 56

Chapter 7: Results and Evaluations 57

7.1) Workflow Engine Applied to a Case Study.....57

7.2) Demonstration (Screen Captures).....58

7.3) Evaluations 63

Chapter 8: Conclusion 64

8.1) Future Works 64

Appendices 65

References 71

Table of Figure

Figure 1: Microsoft Flow 5
Figure 2: If This Then That 6
Figure 3: Camunda BPM Cockpit 7
Figure 4: Alfresco 7
Figure 5: Comparing Table 8
Figure 6: Make Announcement BPMN Diagram 15
Figure 7: Pizza Delivery BPMN Diagram 16
Figure 8: Use Case Diagram 20
Figure 9: Sequence Diagram of View Task 30
Figure 10: Sequence Diagram of Task Execution 30
Figure 11: Sequence Diagram of Execution of Workflow 31
Figure 12: Sequence Diagram of Track workflow 32
Figure 13: Sequence Diagram of Create Flow 33
Figure 14: System Architecture 34
Figure 15: User Model 35
Figure 16: Class Diagram for the Graph Representation of Workflow 36
Figure 17: Workflow Engine Database Diagram 37
Figure 18: Django Channel 40
Figure 19: Illustration of HTTP Web Communication 41
Figure 20: Illustration of HTTP WebSockets Communication 42
Figure 21: JSON-enabled Consumer 43
Figure 22: Software Components 44
Figure 23: BPMN.io web-based BPMN 2.0 Modeler 45
Figure 24: JSON Data Embedded into an XML in a Documentation Tag.....46
Figure 25: BPMN 2.0 Element as an Object-based Class Diagram 48

Figure 26: XML Parser Flow Chart 49
Figure 27: XML Parser's Attributes and Methods 50
Figure 28: Execution of Workflow Template Flow Chart 51
Figure 29: BPMN Flow Graph Representation 52
Figure 30: Workflow Engine Database Diagram of Tables Storing Temporary Data.....53
Figure 31: Exclusive Gateway with Two Outgoing Flows 54
Figure 32: Exclusive Gateway Decisions Represented in Python Dictionary. 54
Figure 33: Exclusive Gateway Decisions Represented in User's Input Form 55
Figure 34: Splitting and joining Parallel Gateway 56
Figure 35: Example of Diagram with Event Handling 56
Figure 36: Pizza Delivery BPMN Diagram 57
Figure 37:User Index Page 58
Figure 38: Create New Workflow Page 59
Figure 39: BPMN Drawing Tools 59
Figure 40: Form Creator (1) 60
Figure 41: Form Creator (2) 60
Figure 42: Task Notification 61
Figure 43: Executing User Task (Fill in Form) 61
Figure 44: Executing User Task (Exclusive Gateway) 62
Figure 45: Executing User Task (Task Submission) 62
Figure 46: Track User's Executing Flow 63

Chapter 1

Introduction

1.1 Motivation

Ina a daily life, one does several activities and handles many events. In other words, one's daily life is governed by a business workflow of those activities and those events. If one can automate one's repeats activities and the routine event handlings, this will greatly improve human productivity.

The aim of this project is to develop an automate business process engine which can executes and monitors the state of activities in a workflow. The workflow engine allocates tasks to different executors while communicating data among the tasks and sending events among participants.

The workflow engine is running in itimemachine.net which is a web application that was also developed for this project.

1.2 Objectives

The objective of this project is to develop a workflow engine that provides users with more efficiency and functionalities by providing users with

- **Visual programming** for composing business flow by drawing BPMN diagram
- **Ability to program services onto diagram** including users' interaction
- **The automation and execution** of user's composed BPMN diagram

With this engine, authorized users will have the ability to:

- Create their own workflow.
- Download the XML script files that generated from the created workflow.
- Download the generated forms from created flows.
- Save and edit an unfinished flow, in order to continue drawing it later on.
- Browse user's XML local files.

- Real-time collaboration with other users.

1.3 Problems Description

In the old days, tasks handling in an organization or individual were done manually which could results in faulty execution, lack of efficiency, and time consuming. This problem has inspired us to develop an automate business process engine in order to help reducing time by automating all the repeated tasks and increasing business productivity.

1.4 Scope of Work

- Doing survey of existing works of workflow management software as well as related apps.
- Web application develop by using Django as our software development framework and ProstgreSQL as the Database Management System.
- Since the workflow engine system needs to really capture the full logic to do an execution, Thus we have to study Business Process Model and Notation (BPMN) which is a business processes, diagrammatic language for modelling.
- Developing a web-based application which allows users to create and edit workflow process diagrams by using BPMN Viewer and Editor, then an engine which automatically generates XML script in order to execute and monitor a workflow process.
- Applying Workflow Engine to test with a real-world as example.

1.5 Contribution

This workflow engine is a new idea. We tried to develop a web application which can connect activities on the web so that people can collaborate more productively. What we have done are listed as follows:

- **BPMN Diagrams for Programming Workflows**

  We created a tool for composing BPMN diagrams embedded with program codes.

- **BPMN Diagrams Transformation to an Executable Flow Graph**

  We developed an algorithm to transform an XML form of BPMN diagram into Python objects of a flow graph.

- **Workflow Interpreter**

  We developed an interpreter to execute a graph form of a BPMN workflow.

Workflow engine was set to be able to more with less and with the purpose to eliminate any risk interfering with efficiency and correctness. It is also beneficial to both companies and individuals since it increase team collaboration, business productivity, and interoperability between services along with automating repeated tasks.

1.6 Thesis Structures

The rest of the thesis are organized as follows:

- **Chapter 2: Related Work**

  Lists and describes existing works which are workflow management system related. This chapter also provides a comparison table of features between Workflow Engine and all those listed related works.

- **Chapter 3: Business Workflow**

  Describes the required background knowledges of business workflow which are Business Process Model and Notation (BPMN), along with the explanation on modelling, executing, and automating a flow.

- **Chapter 4: Software Development**

  This chapter of the thesis was divided into two parts. First part of this chapter will list the requirements of the project, both functional and nonfunctional, use case and its description, and a sample of sequence diagram.

  Second part of this chapter will describe the design of the project. Starting with system architecture, class diagram. Then moving

on to the explanation of graph representation of workflow and lastly, the class diagram for the graph representation.

- **Chapter 5: Web Application Development**

  Describes the tools and knowledges needed for developing both basic and realtime web application.

- **Chapter 6: Workflow Engine**

  Describes the developing the core of this project which is the workflow engine. Explaining the use of BPMN diagram and the handling of the event in a flow.

- **Chapter 7: Results and Evaluations**

  Shows result of the project. The output of the project, the interfaces of the website, and the walkthough guide on how to use this project in the most efficient way.

- **Chapter 8: Conclusion**

  Concluding remark the advantages and limitations of this project. Provides possible future work and the direction that this project could leads to.

Chapter 2

Related Works

In this chapter, we surveyed some existing works which are related to workflow management products, including Microsoft Flow, IFTTT, Camunda BPM, and Alfresco.

2.1 Microsoft Flow

Microsoft Flow is a portable application as in iOS and android that empowers clients to make automated workflows between their most loved applications and services and many more. Microsoft Flow works like IFTTT. For the most part, this application let clients automate document synchronization, alarming, data organization, get notification, and that's only the tip of the iceberg. Microsoft Flow is, for the most part, centered around combinations with Microsoft's own particular business tools, such as Office 365, Dynamics CRM, and PowerApps, and also those utilized as a part of associations, for example, GitHub, and others.

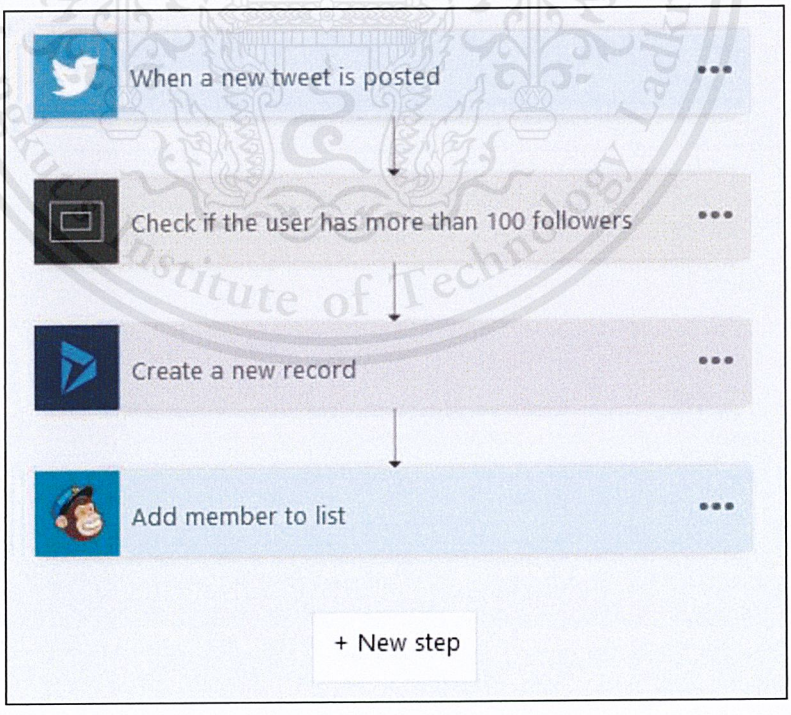


Figure 1: Microsoft Flow

2.2 IFTTT

IFTTT stands for "If This Then That" which is a free online service that let clients make their own particular condition statement by keeping things extremely straightforward as its name by method for illustration. If you go to an eatery and request a burger combo then you will get fries as an afterthought. Fundamentally, IFTTT will computerize activities between your web applications. For example, automatically saving photographs that clients are tagged on Facebook to Dropbox and others. This automate undertaking can likewise be valuable for minimize forms alongside sparing your time.

Figure 2: If This Then That

2.3 Camunda BPM

It is an open source Java based BPM stage utilized basically to mechanize Business Process Model and Notation (BPMN) 2.0 procedures. Camunda working procedure initially are making BPMN handle graphs and DMN choice tables and then Camunda will transform client model into an executable applications, Next camunda tasklist will let end clients can take a shot at the errands doled out. At that point all data on finished exchanges can be stored in Camunda BPM and saw by means of Camunda cockpit. It is worked around the procedure motor segment. Still, there are sure constraints to the utilization of BPMN 2.0 components in Camunda. A few components are reliant on particular demonstrating criteria, neglecting to fulfill which doesn't deliver the normal yields.

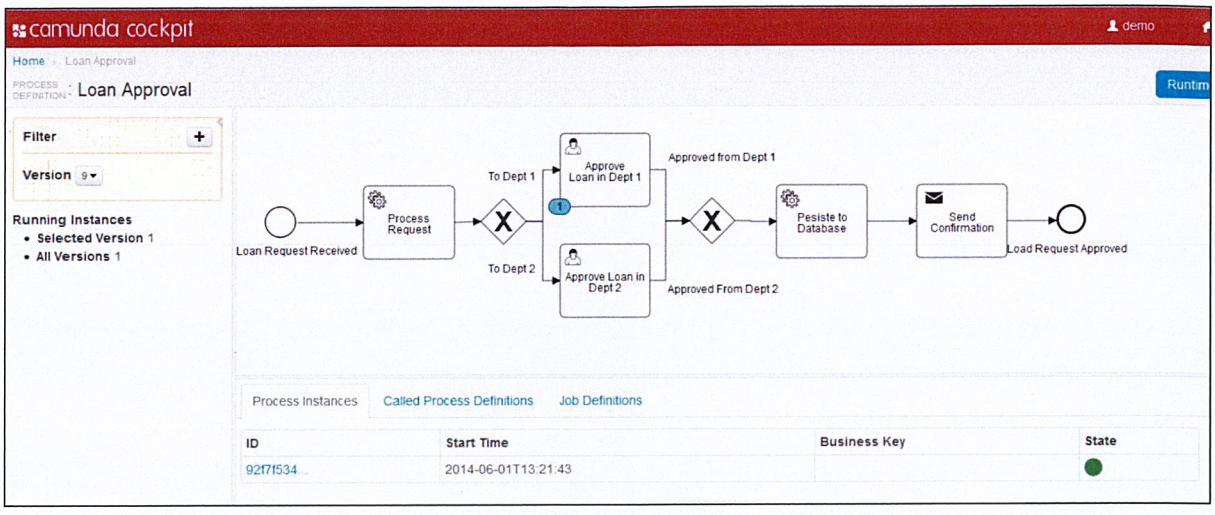


Figure 3: Camunda BPM Cockpit

2.4 Alfresco

Alfresco is a free system for Microsoft Windows and Unix-like operating systems. Alfresco's product initially focused on document management. Alfresco is not a Web Content Management system. It is an Enterprise Content Management system and Document Repository. Alfresco provides point-and-click forms editor, can automated document creation, and is compatible with other sharing systems such as Google Drive.

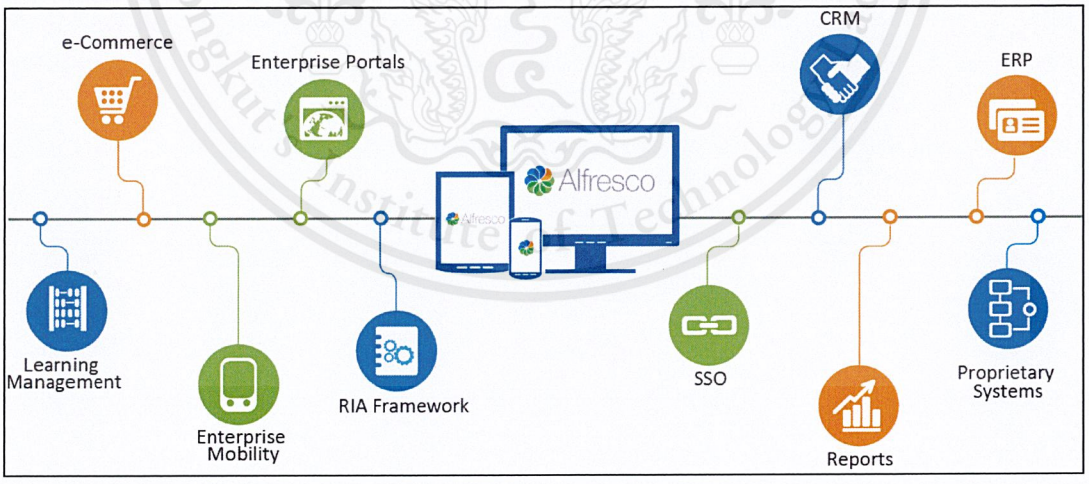


Figure 4: Alfresco

2.5 Comparing table between Our Engine and other related work.

Features / Products	Microsoft Flow	IFTTT	Camunda BPM	Alfresco	Our Engine
Support BPMN 2.0			✓	✓	✓
Web-based application	✓	✓			✓
Document Attachment			✓		✓
Real-time Notification		✓			✓
Generate Form			✓	✓	✓

Figure 5: Comparing Table

*Premium services and organization features is available for only premium user

Chapter 3

Business Workflow

In this chapter we describe background knowledges of business workflow. What it is, how it works, and how we model it.

3.1 Modelling of Business Workflow

A business Workflow or just Workflow is a progression of works which illustrates a work process. A workflow always involves at least two persons and works with procedural rules specifically set for each work process.

There are several formats to draw a business process. The most use and effective one is the swimlane format due to its highlighting on relevant variables and it is very easy to understand without the need of much background knowledges.

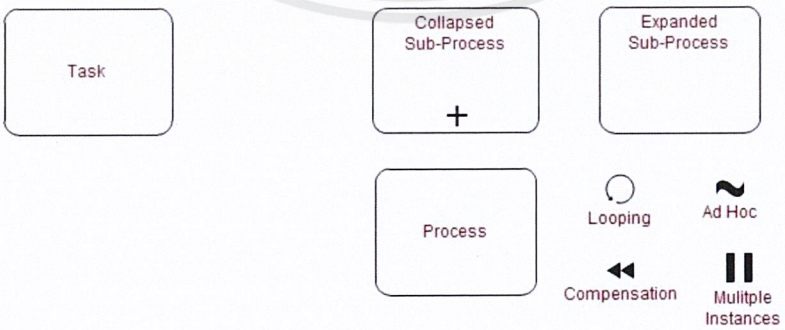
3.2 Business Process Model and Notation (BPMN)

The primary goal of BPMN is to provide a standard notation readily understandable by all business stakeholders. BPMN serves as a common language, bridging the communication gap that frequently occurs between business process design and implementation

A BPMN model consists of simple diagrams constructed from a limited set of graphical elements. For both business users and developers, they simplify the understanding of business activities' flow and process.

3.2.1 BPMN Notations

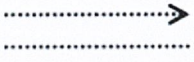
- **Flow Objects**: A Business Process Diagram has a small set of (three) core elements, which are the Flow Objects, so that modelers do not have to learn and recognize a large number of different shapes. The three Flow Objects are

Events	An Event is represented by a circle and is something that "happens" during the course of a business process. These Events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results.
 Start	The Start event show which event cause the process to start.
 Intermediate	This notation stands for a status that is reached in the process and that is modeled explicitly. They are used infrequently, but can be useful.
 End	The End event signifies the status reached at the end of the process path.
Activities	An Activity is represented by a rounded-corner rectangle and is a generic term for work that company performs. An Activity can be atomic or non-atomic (compound). The types of Activities are: Task and Sub-Process. The Sub-Process is distinguished by a small plus sign in the bottom center of the shape.
	

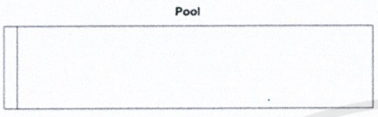
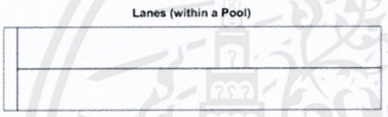
Gateways	A Gateway is represented by the familiar diamond shape and is used to control the divergence and convergence of Sequence Flow. Thus, it will determine traditional decisions, as well as the forking, merging, and joining of paths. Internal Markers will indicate the type of behavior control.
 Exclusive Gateway	When splitting, an Exclusive Gateway routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.
 Event-based Gateway	The Event-based Gateway is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.
 Parallel Gateway	The Parallel Gateway is used to split the sequence flow, and all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.
 Inclusive Gateway	The Inclusive Gateway when splitting, one or more branches are activated. All active incoming branches must complete before merging.
 Complex Gateway	This Complex Gateway is used for merging and branching behavior which is not captured by other gateways.
 Exclusive Event-based Gateway (instantiate)	With this gateway, each occurrence of a subsequent event starts a new process instance.

 Parallel Event-based Gateway (instantiate)	With this gateway, the occurrence of all subsequent events starts a new process instance.

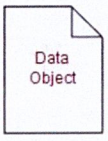
- **Connecting Objects**: The Flow Objects are connected together in a diagram to create the basic skeletal structure of a business process. There are three connecting object that provides this function, which are

Sequence flow	A Sequence Flow (or Control Flow) is represented by a solid line with a solid arrowhead and is used to show the order (the sequence) that activities will be performed in a Process. Note that the term "control flow" is generally not used in BPMN
Message flow	A Message Flow is represented by a dashed line with an open arrowhead and is used to show the flow of messages between two separate Process Participants (business entities or business roles) that send and receive them. In BPMN, two separate Pools in the Diagram will represent the two Participants.
Association 	An Association is represented by a dotted line with a line arrowhead and is used to associate data, text, and other Artifacts with flow objects. Associations are used to show the inputs and outputs of activities.

- **Swimlanes**: Many process modeling methodologies utilizes the concept of swimlanes as a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities. BPMN supports swimlanes with two main constructs, which are

Pool  	A Pool represents the process participants. It is also acts as a graphical container for partitioning a set of activities from other Pools.
Lane  	A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organize and categorize activities or to show that certain units/roles are responsible for performing specific process steps.

- **Artifacts**: Any number of Artifacts can be added to a diagram as appropriate in the context of the business processes being modeled. The current version of the BPMN specification pre-defines only three types of BPD Artifacts, which are

Data Object 	A data Objects are a mechanism to show how data is required or produced by activities. They are connected to activities through Associations.
Group 	A Group is represented by a rounded corner rectangle drawn with a dashed line. The grouping can be used for documentation or analysis purposes, but does not affect the Sequence Flow.
Annotation 	An annotation is a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram.

BPMN Diagram examples

Diagram #1: Make Announcement of a news posting

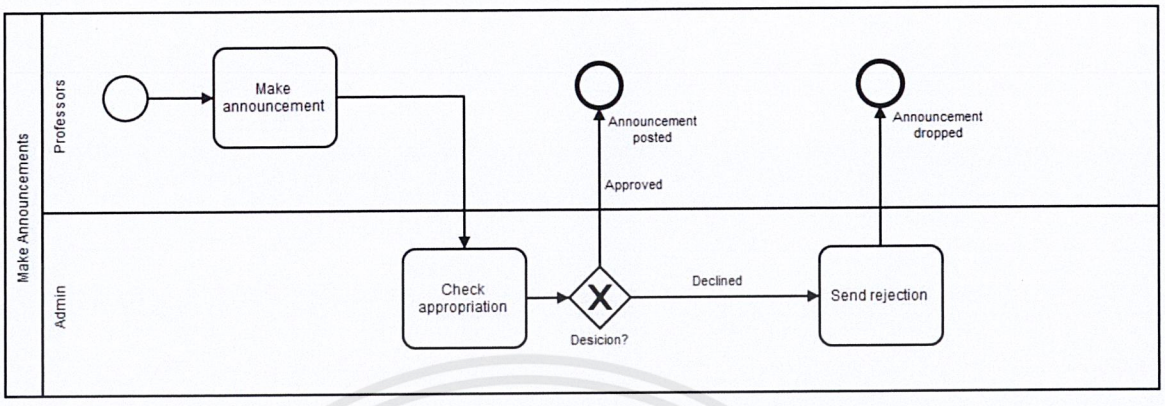


Figure 6: Make announcement BPMN diagram

Description:

This BPMN diagram has one start event and two different end events with three tasks and one exclusive gateway. The output or the ending of this diagram depends on the admin's decision at the exclusive gateway.

The process of this workflow starts when the professor wants to make an announcement. The professor announce something onto the website, after that the admin will check the appropriation of that post and decide if it appropriate or not. If yes, the announcement will be posted. If not, the announcement will dropped.

Diagram #2: Pizza Ordering

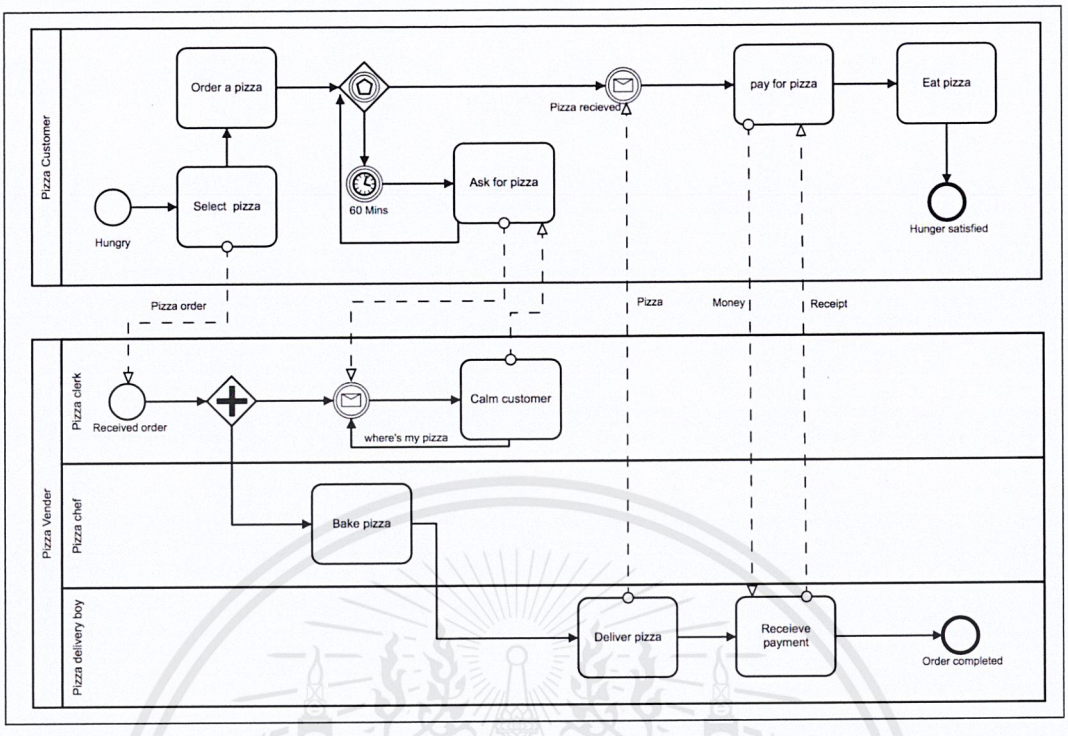


Figure 7: Pizza delivery BPMN Diagram

Description:

This Pizza Ordering workflow consists of two pools which are customer and pizza vendor. The main focus in this diagram is the two gateways, one is the exclusive event-based gateway in the customer pool, it has two events that act as a trigger which are pizza received and still no pizza after sixty minutes. Another gateway is parallel gateway. Both of the branches which leaving this parallel gateway will activate simultaneously, means that while the chef is baking the pizza, the clerk is also waiting for customer to call and ask for their pizza. If the customer never called and ask for their pizza, it could only means that the pizza vendor delivered the pizza within sixty minutes. Customer and pizza vendor are communicating to each other via a message flow(the dash lines) that act as a messenger which send messages between them. Though this diagram has two end events, it doesn't mean that it has two different ending like the diagram mentioned above. Due to its two pools, both customer and pizza vendor has their own end event.

noted that this parallel gateway was spilt but never join

3.2.2 Execution/ Interpretation

The basic concept of workflow states that the execution of a flow is that when "one step has been completed successfully and the next step can begin". Most people and business groups can consider a "workflow" as the everyday steps that they embrace with a specific end goal to finish their ventures. A linear or a more lateral approach is advised in order to execute a workflow since each task or process has a specific before and after tasks. So if we look at it in a graph or a linear form, it is easier to optimise those tasks and improve work performance.

3.2.3 Workflow Automation

Workflow automation is to eliminate or replace the manual and paper-based task that occurs in our everyday life. By automating the workflow, users or employees can focus more on their work other than the little details that support it. They invest less energy and time in redundant work and more on their ability. With process examination, your group can screen, break down, and respond to the information behind procedures. They can react speedier to client needs and market situations.

Advantages of automating a workflow:

- Improved coordinated effort and better correspondence.
- More effective operations.
- Extended portability.

Chapter 4

Requirements Analysis

4.1 Requirements

4.1.1 Functional Requirements

- The system provides users with registration system.
- The system will store user's information which will be visible in user profile.
- The system allows users to edit and modify their profile.
- The system gives user a choice between use or customise the existing workflows or create their own new one.
- The system lets users save, view, update, or delete their workflows.
- The system lets users download XML script file that generated from the created workflow.
- The system will generate a form, according to user's created flow.
- The system lets users keep track of their executing workflow.
- The system lets both users and admin choose to execute a flow and tasks within a flow.
- The system lets user request task ID.
- The system lets administrator CRUD (create, read, update, and delete) both flows and users.
- The system lets administrator approve user's workflows.

4.1.2 Non-Functional Requirements

- The workflows are based on BPMN 2.0.
- The system shall has real time responding.
- The system shall be a web application in which users can interact through a web browser.
- The system shall support the following web browsers: Firefox, Google Chrome, Internet Explorer, and Safari.
- The system shall not allow unauthorised users to use any feature on the website, except for registration.

4.1.3 Use Case Modeling

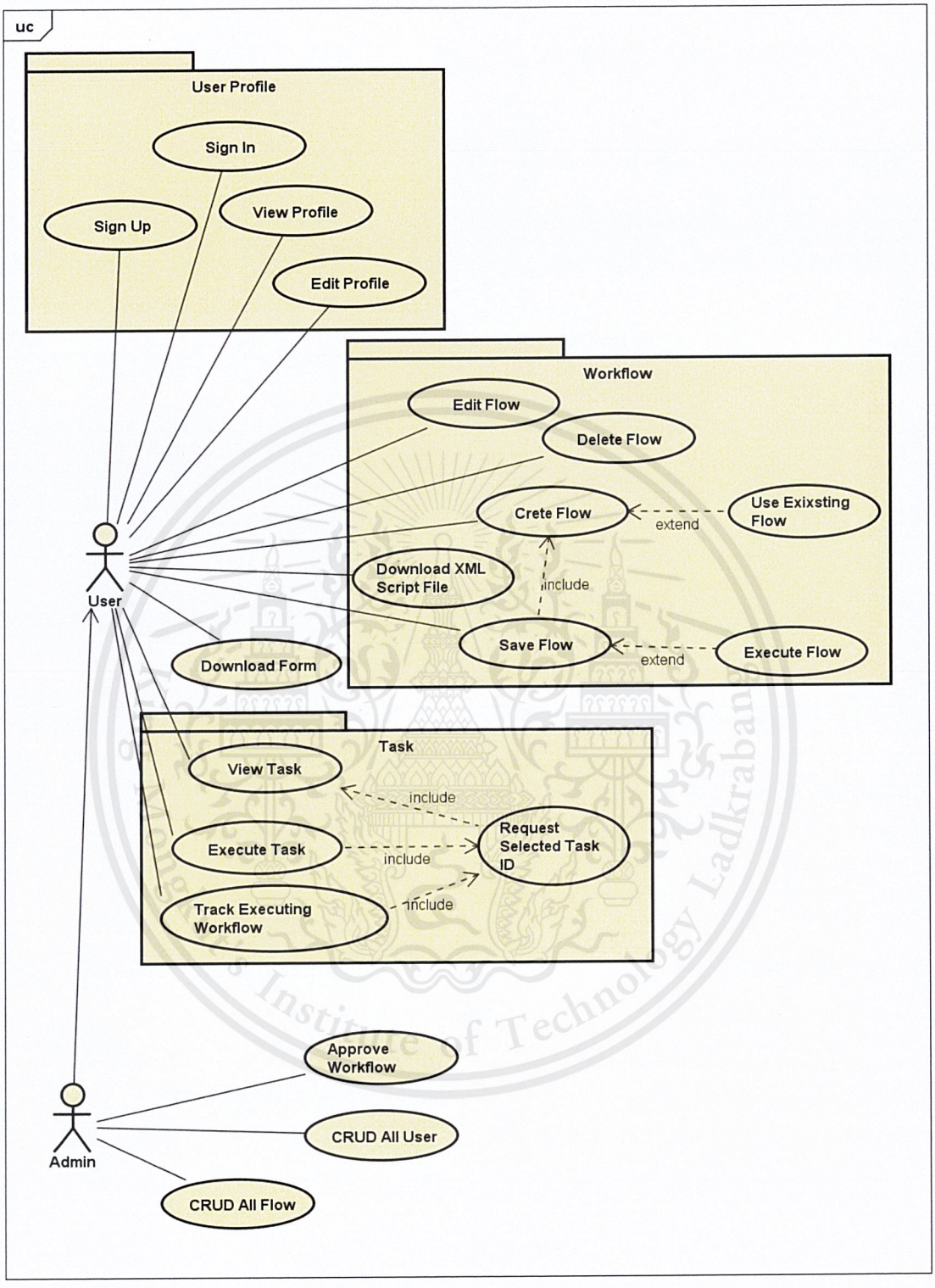


Figure 8: Use Case Diagram

Use Case Description

Use Case	**Sign Up**	
Primary Actor	User	
Flow of Events	**Actor Input**	**System Response**
	1) Select sign up option	
		2) Show sign up page
	3) Fill in the required information	
		4) Registered user

Use Case	**Sign In**	
Primary Actor	User	
Precondition	Have an account	
Flow of Events	**Actor Input**	**System Response**
	1) Fill in username and password	
	2) Select sign in option	
		3) Show main page
Alternative Flow	**Condition:** Wrong username or password	
		4) Request for another username or password

Use Case	View Profile	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	Actor Input	System Response
	1) Select view profile	
		2) Show user's profile page

Use Case	Edit Profile	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	Actor Input	System Response
	1) Select view profile	
	2) Select edit profile	
		3) Show edit profile page
	4) Fill in the required information	
	5) Select save	
		6) Save new user's profile

Use Case	Use Existing Flow	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	Actor Input	System Response

	1) Choose one of the provided flow	
		2) Show draw diagram page (with already drawn diagram)

Use Case	**Create Flow**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Select create new flow	
		2) Show draw page
	3) Draw diagram with provided tools	
	4) Select save	
		5) Save the created to user's profile
	6) Select download XML file	
		7) Save the file to user's computer

Use Case	**Save Flow**	
Primary Actor	User	
Precondition	Is an authorized user, just created a flow	
Flow of Events	**Actor Input**	**System Response**
	1) Choose save flow	

		2) Save the created workflow to user's profile onto the website

Use Case	**Edit Flow**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Choose his/her own saved flow	
		2) Show draw page (with saved diagram)
	3) Edit the diagram with provided tools	
	4) Choose save flow	
		5) Save the created to user's profile

Use Case	**Delete Flow**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Choose his/her own saved flow	
		2) Show draw page (with saved diagram)
	3) Select delete flow	

		4) Delete the flow from user's profile

Use Case	**Download XML file**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Choose his/her own diagram or existing one	
		2) Show draw page (with the chosen diagram)
	3) Select download XML file	
		4) Save diagram's XML file to user's computer

Use Case	**Execute Flow**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Select an available flow	
	2) Select execute (start)	
		3) Start the workflow

Use Case	**Download Form**
Primary Actor	User

Precondition	Is an authorized user, finish creating a flow	
Flow of Events	**Actor Input**	**System Response**
		1) Generate a form from the flow
	2) Select download form	
		3) Save the form to user's computer

Use Case	**View Task**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Select Task page	
		2) Show task page containing task that associated with current user

Use Case	**Execute Task**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Choose an available flow	
		2) Show diagram page
	3) Choose a task within the selected flow	
	4) Select execute	

		5) Begin progress

Use Case	**Track Executing Flow**	
Primary Actor	User	
Precondition	Is an authorized user	
Flow of Events	**Actor Input**	**System Response**
	1) Choose an available executing floe	
		2) Show process of the flow
	3) Select specific task	
		4) Show selected task details

Use Case	**Request Selected Task ID**	
Primary Actor	Admin	
Precondition	-	
Flow of Events	**Actor Input**	**System Response**
	1) Choose an available flow	
		2) Show draw page (with selected diagram)
	3) Select a task	
	4) Request selected task ID	
		5) Show the selected task ID

Use Case	**Approve Workflow**

Primary Actor	Admin	
Precondition	-	
Flow of Events	Actor Input	System Response
	1) Select a flow to approve among the list of user's flows	
	2) Select approve	
		3) Flow shows in approved flow list

Use Case	CRUD All User	
Primary Actor	Admin	
Precondition	Website consists of at least one user.	
Flow of Events	Actor Input	System Response
	1) Select manage user	
		2) Show users page
	3) Select actions	
	4) Select save	
		5) Save action

Use Case	CRUD All Flow	
Primary Actor	Admin	
Precondition	-	
Flow of Events	Actor Input	System Response
	1) Choose an available flow	

		2) Show draw page (with saved diagram)
	3) Update or Delete the diagram	
	4) Select save	
		5) Save action
Alternative Flow	**Condition:** Admin choose create new workflow	
		6) Show draw page
	7) Draw diagram with provided tools	
	8) Select save	
		9) Save action

4.1.4 Sequence Diagrams

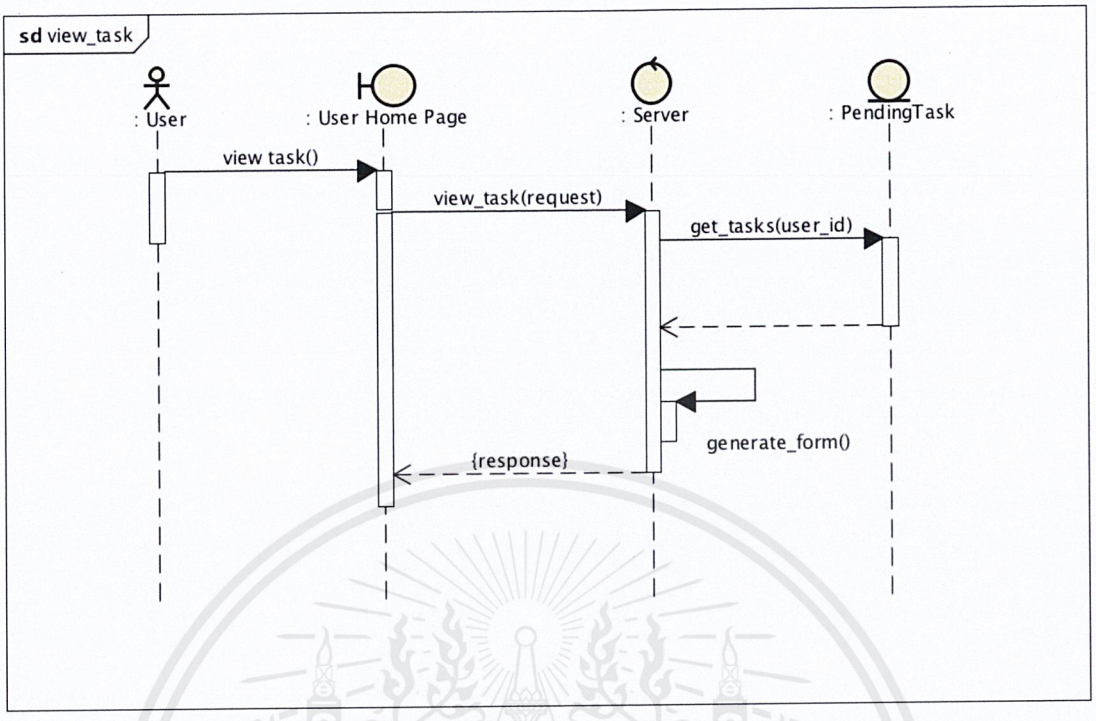


Figure 9: View Task

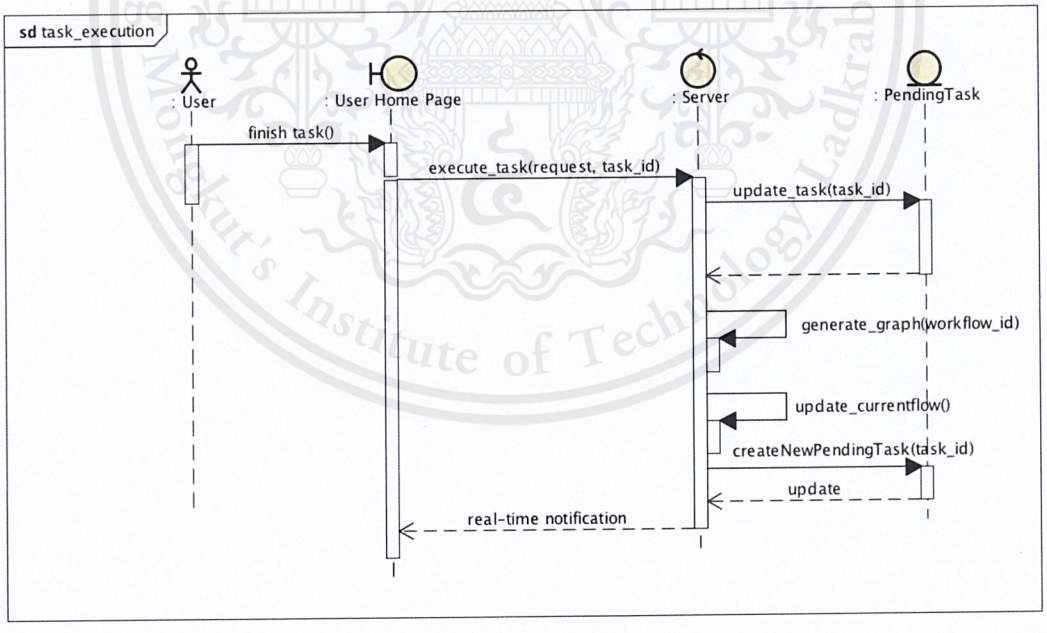


Figure 10: Task Execution

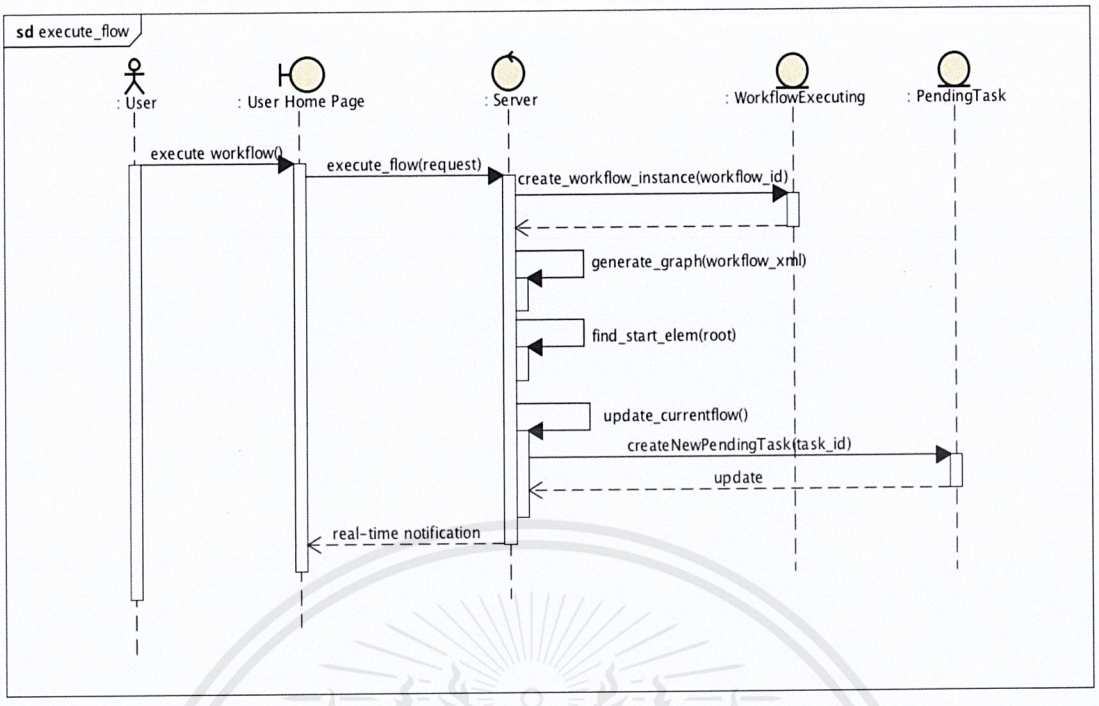


Figure 11: Execution of Workflow

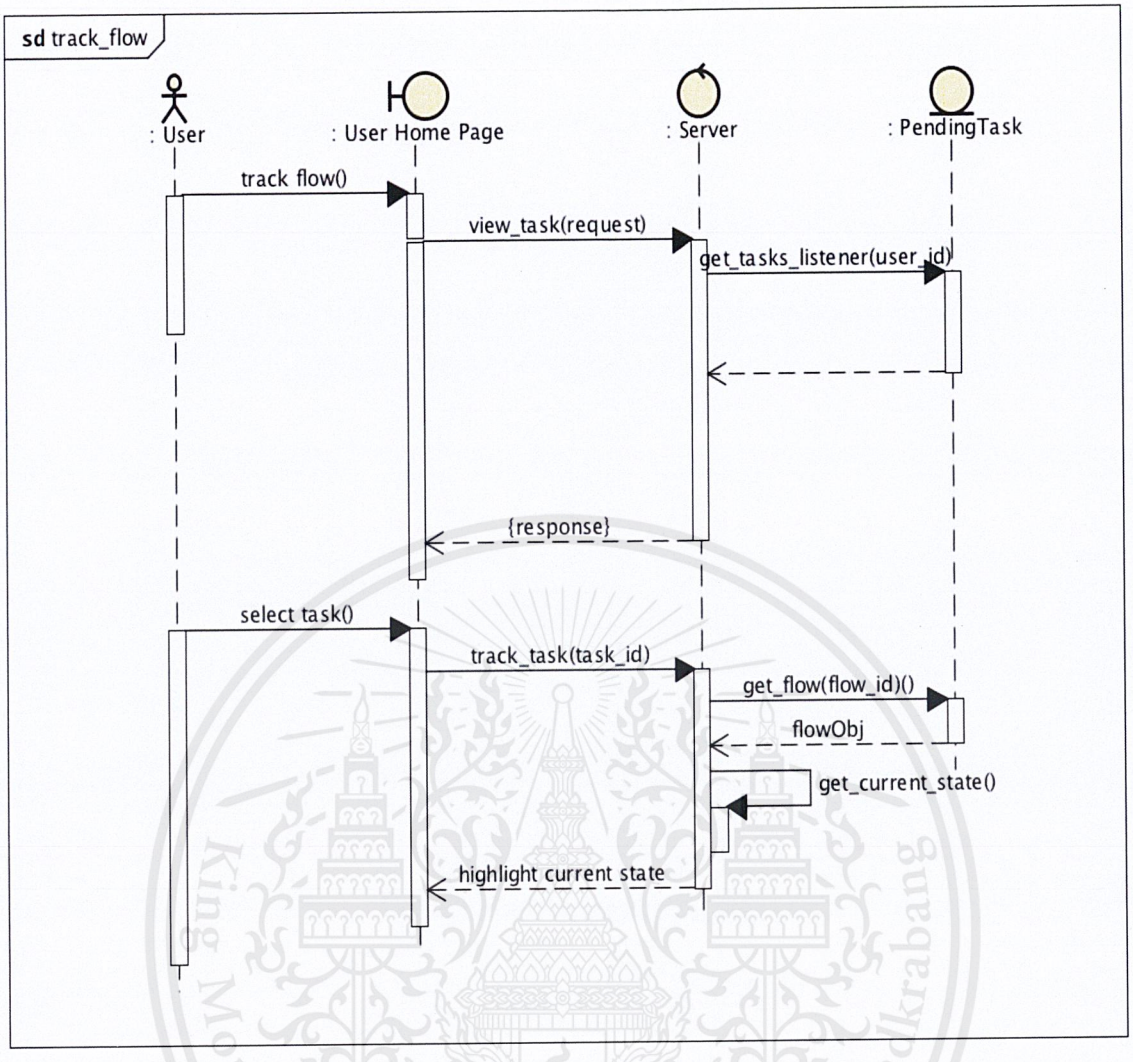


Figure 12: Track Workflow

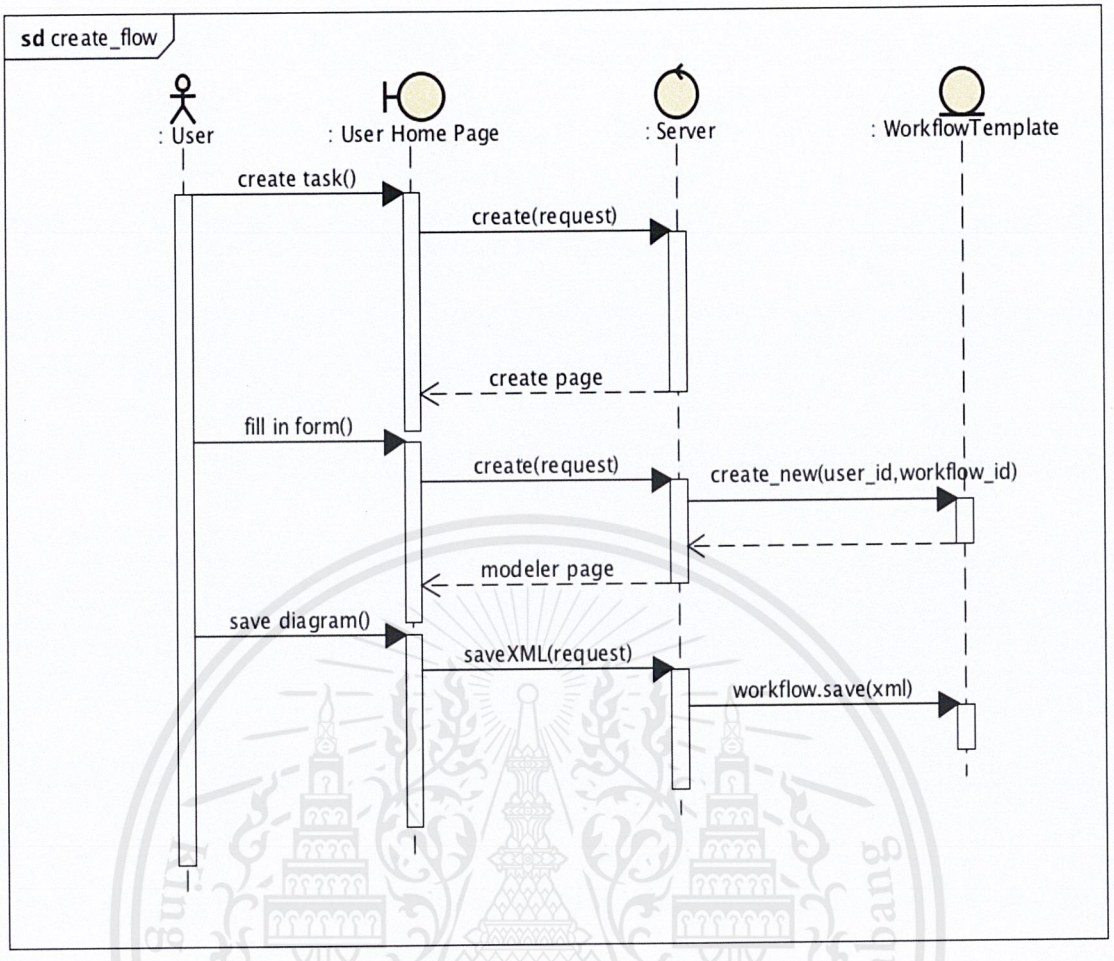


Figure 13: Create Flow

4.2 Designs

4.2.1 The system Architectures

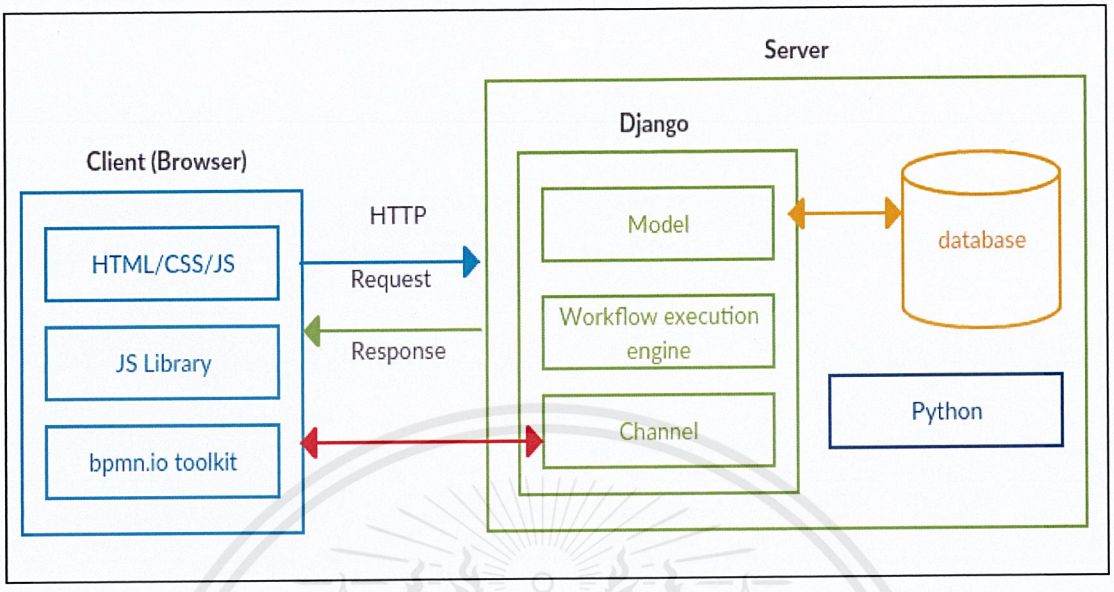


Figure 14: System Architecture

The whole system composes of client and server sides where the client sends requests to resources and the server responds directly to the request by using its own resources.

- **Client-side**
  - It has mostly to do with the user interface which the user interacts. We use HTML, JavaScript, CSS for developing our website along with bmpn.io for the diagram drawing tool.
- **Server-side**
  - We use Django which is a high-level Python Web framework to implement models, workflow execution engine. Also we make use of channel library to make Django capable to handle more than just simple HTTP requests, including WebSockets and HTTP2. This side interacts with the Postgres database.

4.2.2 Class Diagram for the user model

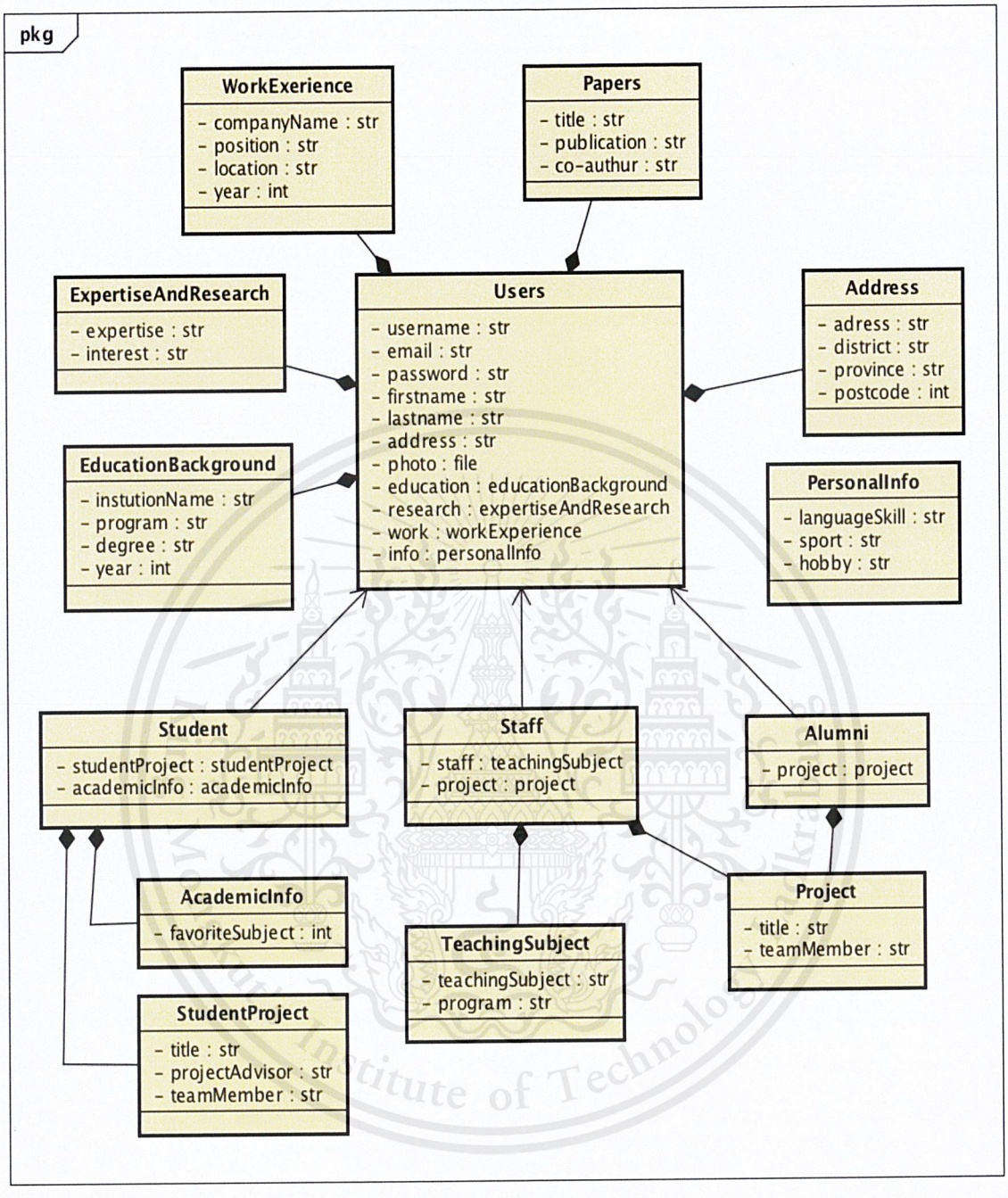


Figure 15: The user Model

4.2.3 Class Diagram For the Graph Representation

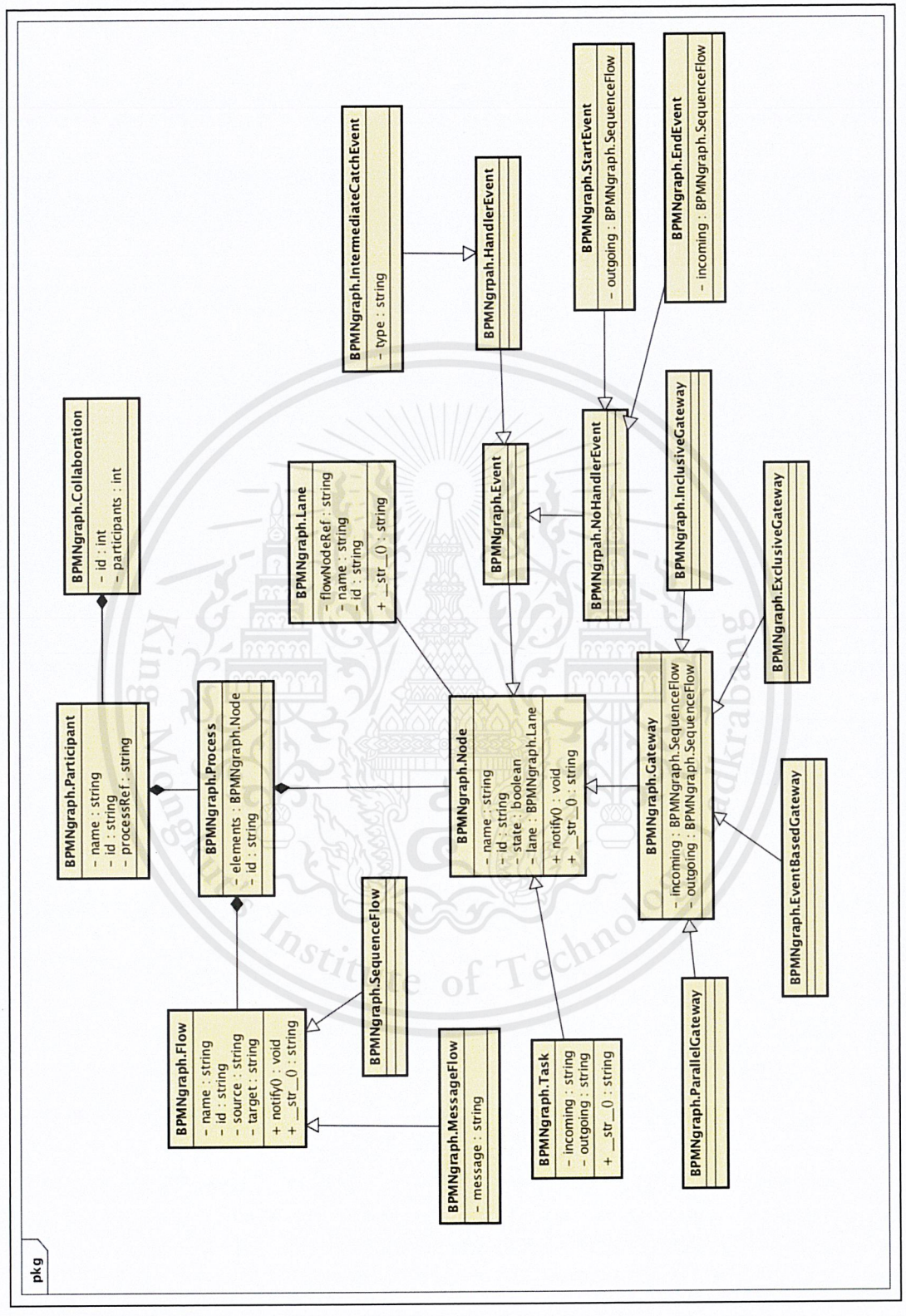


Figure 16: Class Diagram for the Graph Representation of workflow

4.2.4 Workflow Database Diagram

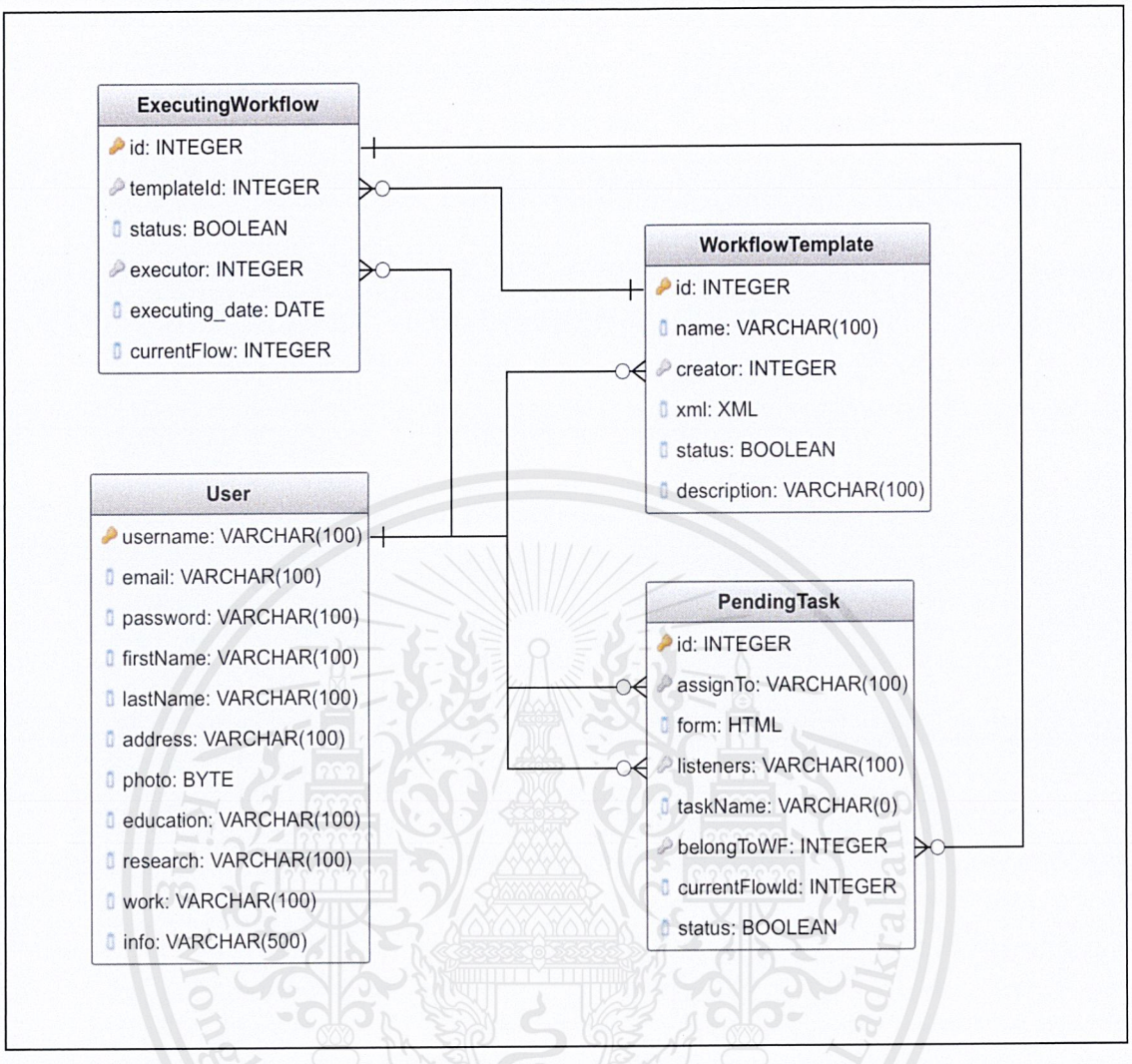


Figure 17: Workflow Engine Database Diagram

Chapter 5

Web Application Development

In this chapter, we describe the software tool at the server side that we used to develop the workflow engine.

5.1 Django Web Framework

Django is a high-level Python web framework which supports fast and clean web development created by experienced engineers, it deals with a significant part of the web development where you can concentrate on composing your application without expecting to rehash the wheel. Django is an open source software.

Django's good features:

- Fast
- Fully loaded
- Secure
- Scalable
- Versatile

Django Model Template and View (MTV)

Django loosely follows the MVC design pattern that stands for Model,View, and Controller. Nevertheless Django uses its own software design pattern. It is often referred as MTV framework, where

- **M stands for " Model "**
  - This model layer includes everything about the data. For instance, how to access to data, how to derive it, and the relationships between the data.
- **T stands for " Template "**
  - Template is the presentation layer. This layer contains presentation related decisions of how something should be displayed on a Web page.

- **V stands for " View "**
    - View is the business logic layer. This layer holds the logic that accesses the model and accords to the appropriate templates. Generally speaking, it is the bridge between models and templates.

5.2 Django Channel

Channel helps Django do more than just HTTP request as it changes the way Django work to be event oriented. Channel is "*an ordered, first-in first-out queue with message expiry and at-most-once delivery to only one listener at a time.*". We can say that Channels is a Django extension which adds a new layer that allows two important features:

- WebSocket handling
- Background tasks

Channel works like a task queue where messages get pushed onto the channel by producers. Inside a system, we recognize channels remarkably by a name string - you can send to any named channel from any machine associated with a similar channel backend. On the off chance that two distinct machines both write to the http.request channel, they are composing into the same channel.

Channels allows Django to run in a "channel mode", swapping out the request and response cycle with the channel-based architecture. There is a so called "interface server" who knows how to handle requests using different protocols. The interface server accepts the request and modify it into message, then passes the message onto a channel. Consumers process these queues and write back response on response channels. The interface server send back the responses.

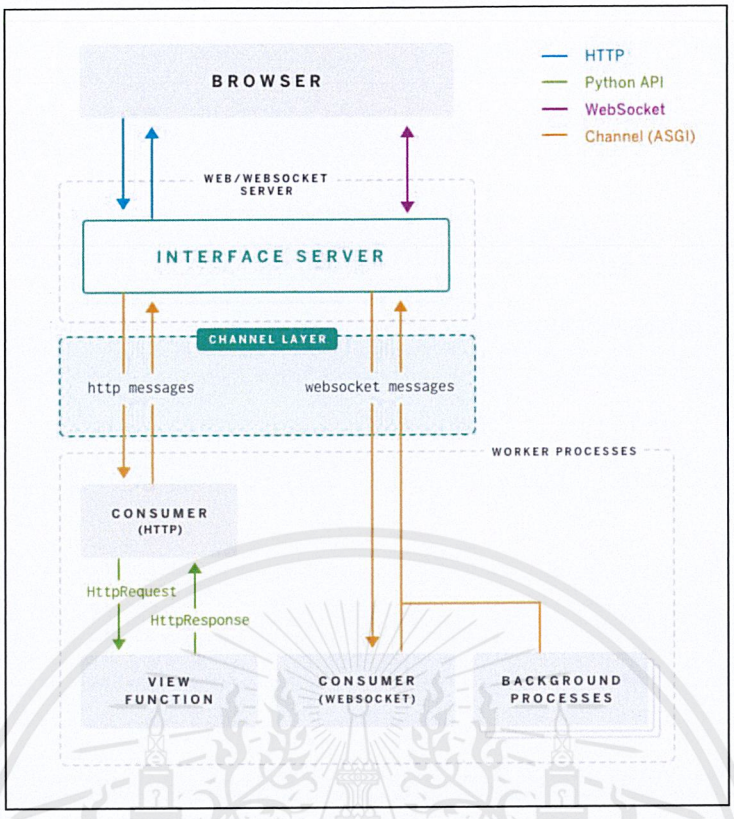


Figure 18: Django Channel

5.3 Websocket

Websocket is a communication protocol, providing full-duplex communication channel over a single connection. It is designed to be implemented on a web browser and a web server, but it can be used by any client. It allows communication between a browser and a web server.

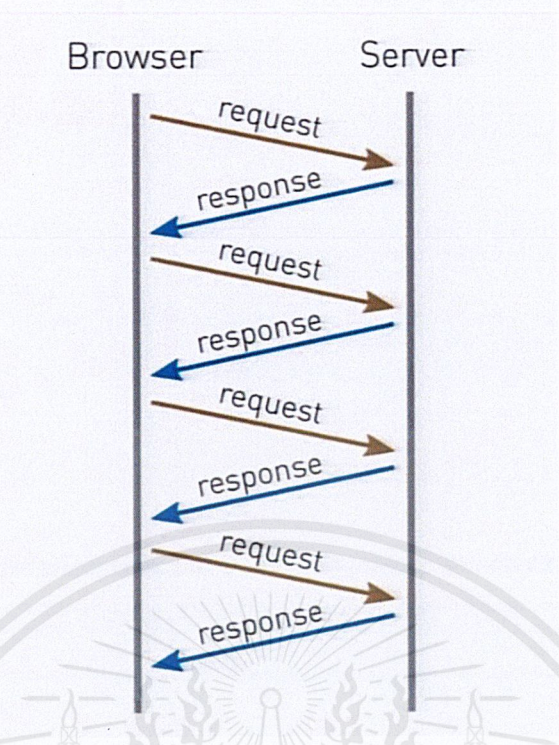


Figure 19: Illustration of HTTP Web Communication

Generally, when client requests for a web page, the web server receives an HTTP request and after it acknowledges this request, the web server sends back the response. However, in many practical cases and some application which require rapid responses or real time interactions or display stream of data such as stock prices, movie ticket sales, working tasks, and so on; this response could be dry out and even not accurate compare to its original data on the web server by the time the web server sends (renders) the page to the client. In order to get the most accurate, up-to-date, real-time data, user is required to refresh the web page manually; but that is obviously not a good way to solve the problem.

For Websocket, HTTP handles connections by whenever client made a request, a port or socket was opened, data was transferred, and the port was closed. This closing and opening require using disk space.

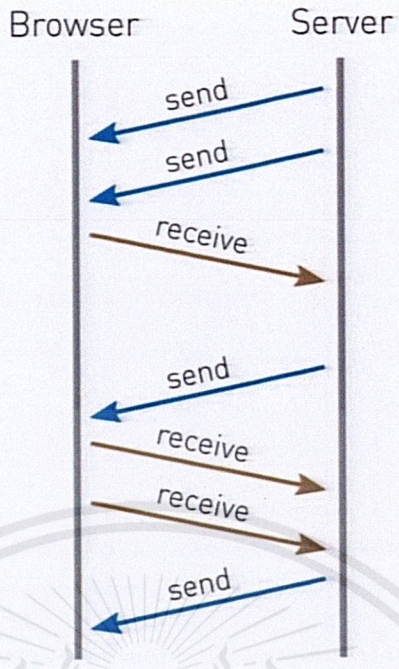


Figure 20: Illustration of WebSockets Communication

WebSockets is a representation of the new evolution of communications between client and server, with an operation of sockets by bidirectional communications channels over the web. This technology provides a standard that we can use to develop web applications with real-time ability and scalable as well as to remove overhead and complexity from Polling, Long-Polling, and Streaming.

Channel has class-based consumers, similarly to Django class-based views. There are two WebSocket generic consumers and this part of the thesis will be focusing on only one of them which is the basic WebSocket generic consumer.

The basic WebSocket generic consumer has basic routing method and a subclass that serializes messages using JSON.

```
class ChannelConsumer(websockets.JsonWebsocketConsumer, EventListenerMixin):
    """
    Manages the channels states through a websocket.
    """
    strict_ordering = False
    event_listener_category = "channels"

    def connection_groups(self, **kwargs):
        return ["chat"]

    def receive(self, message, **kwargs):
        self.run("pre_receive")
        self.group_send(self.connection_groups()[0], message)
        self.run("post_receive")
```

Figure 21: JSON-enabled consumer

Django Channel is incorporated in our web application with the purposes of automating the "new task" for each users and a community billboard for every users to share ideas and work together.

5.4 Software Development Tools Used in This Project

- Client-side Web Development
    - HTML/CSS/JS
- Python programming language
- Django
    - An open source web application framework which is written in Python. This web framework is a set of components that helps developers to develop websites faster and easier
- PostgreSQL
    - An open source object-relational database system

Chapter 6

Workflow Engine

In this chapter, we shall explain how the Workflow Engine is developed.

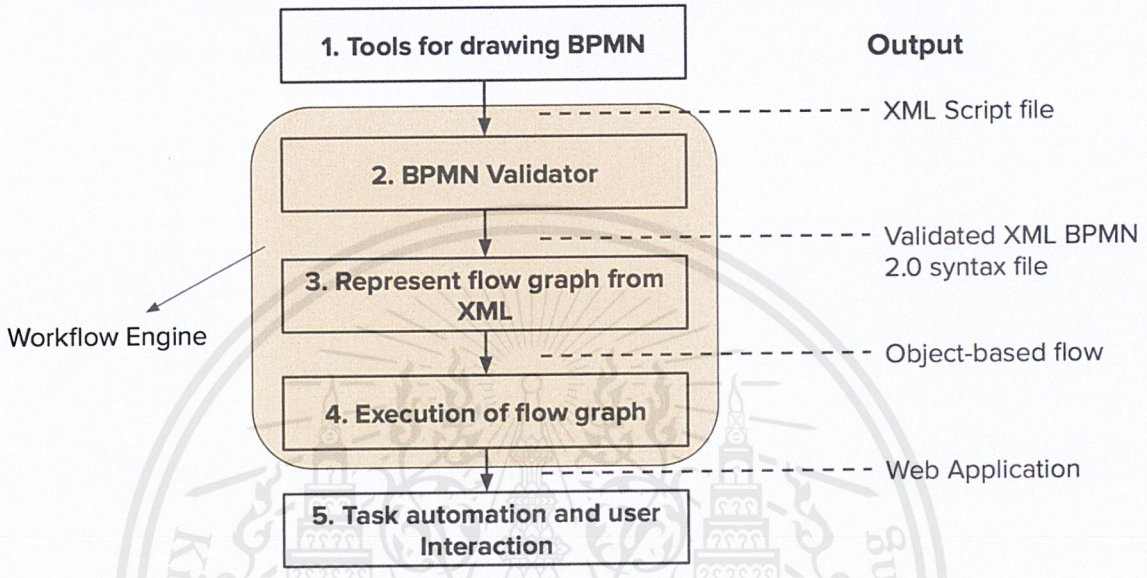


Figure 22: Software Component

Our workflow engine architecture consists of three essential components, an XML/BPMN 2.0 format validator, an XML parser, and an executor of a program flow. For BPMN Validator, we use lxml library to check if the syntax of BPMN 2.0 xml file matches the specification xsd file or not. If the input file contains any syntax or format error, the function returns false.

6.1 The Tool of Drawing BPMN Diagrams

For BPMN 2.0 diagram modeler and viewer, we adopt the bpmn.io, lightweight toolkit for drawing a BPMN 2.0 diagram on the web and generates the XML data model from the created diagram. It can be integrated into any web application to create and render BPMN 2.0 diagrams. This tool allows us to customize and extend the modeler to suit our project, while properties panel and element's document were implemented as an extension to it by ourselves.

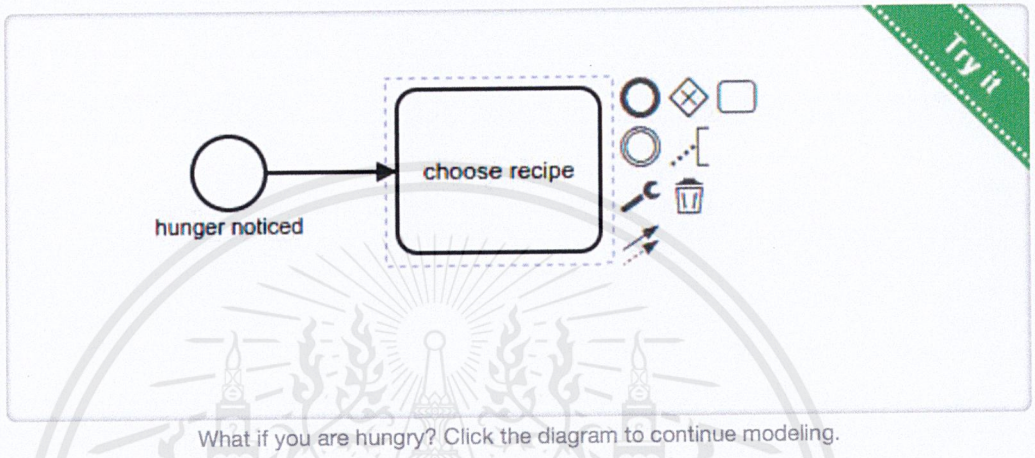


Figure 23: Bpmn.io web-based BPMN 2.0 modeler

In this project, we decided not to use the the full version of this open source drawing tool because it is implemented in node.js architecture and requires complex additional setup as we plan to use Django for development at the server. Instead we use bpmn-js-seed which is a simple project based on HTML and JavaScript which is easier to deploy in our server and does not require any additional setup.

Original bpmn-js-seed Features	Extensions
BPMN 2.0 diagram Viewer	Properties Panel
BPMN 2.0 diagram Modeler	Element's stored document
Generated XML from BPMN 2.0 diagram	Browse XML file from local
	Save diagram as XML file and SVG

6.1.1 Form creation

Form creator is a digital form that allows the user to create as well as customize their own form by drag and drop elements such as header, button, date field, paragraph, text field, and selection provided in the right side of a form creator page.

After a form is created and saved, it will be embedded into the xml file by converting json to string then appending it into a documentation tag which is a hidden tag. For an example; <documentation> JSON data of a created form </documentation>

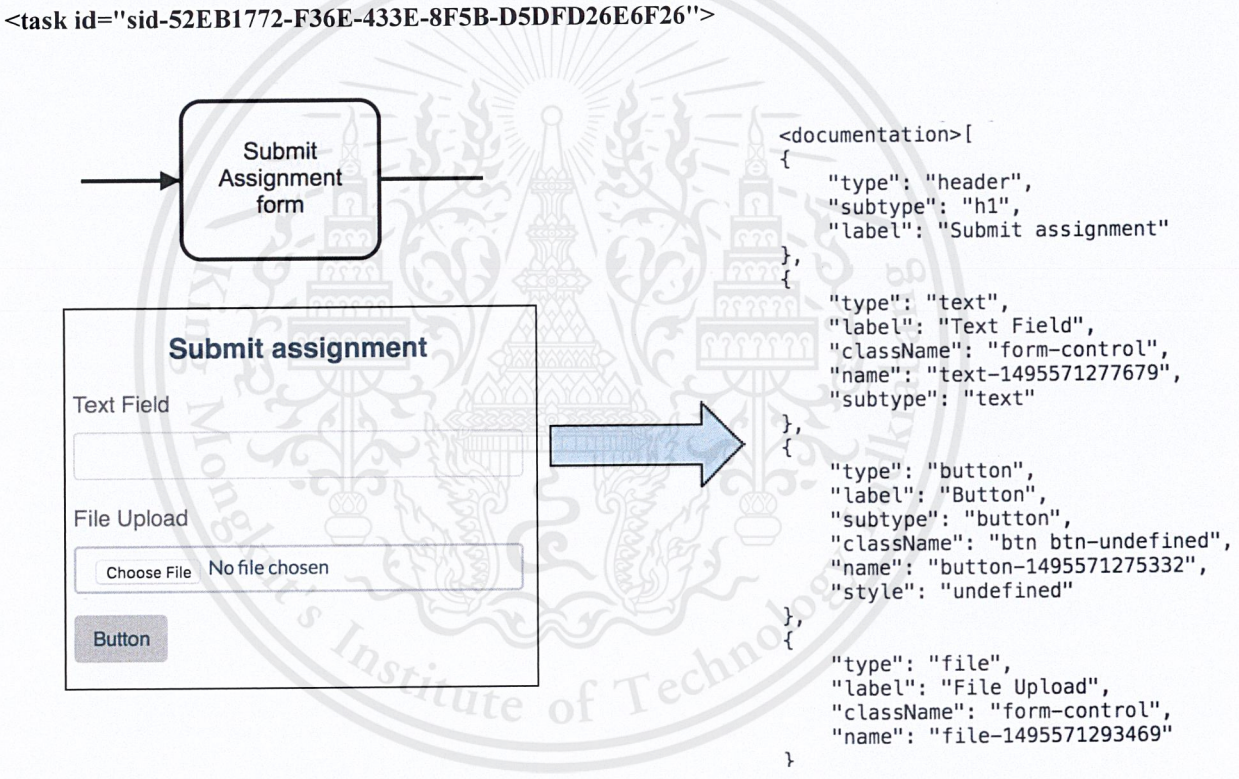


Figure 24: JSON data embedded into an xml in a documentation tag

6.2 Converting BPMN XML to its Graph Representation

In an early stage of the development, we decided to implement our first version of workflow engine as a stand-alone application using Python as it is easier and more convenient for testing and subdivide the whole piece of big project into smaller parts.

In this chapter, an XML parser will be described first. In order to access and traverse through an XML file, it needs to be in a parse tree element format which later can be mapped into an equivalent object-based class to create an instance of that element. This process is handled by an XML parser by mapping each XML element into an object, starting from the root of the tree until reaching its leaves. The equivalent object-based Python class of BPMN 2.0 elements are illustrated in Figure 25:

- Collaboration
- Participant
- Node (Activity)
- Lane
- StartEvent
- EndEvent
- IntermediateCatchEvent
- InclusiveGateway
- ExclusiveGateway
- EventBasedGateway
- ParallelGateway
- Process
- Sequence Flow
- Message Flow

These objects can be represented by the following class diagram:

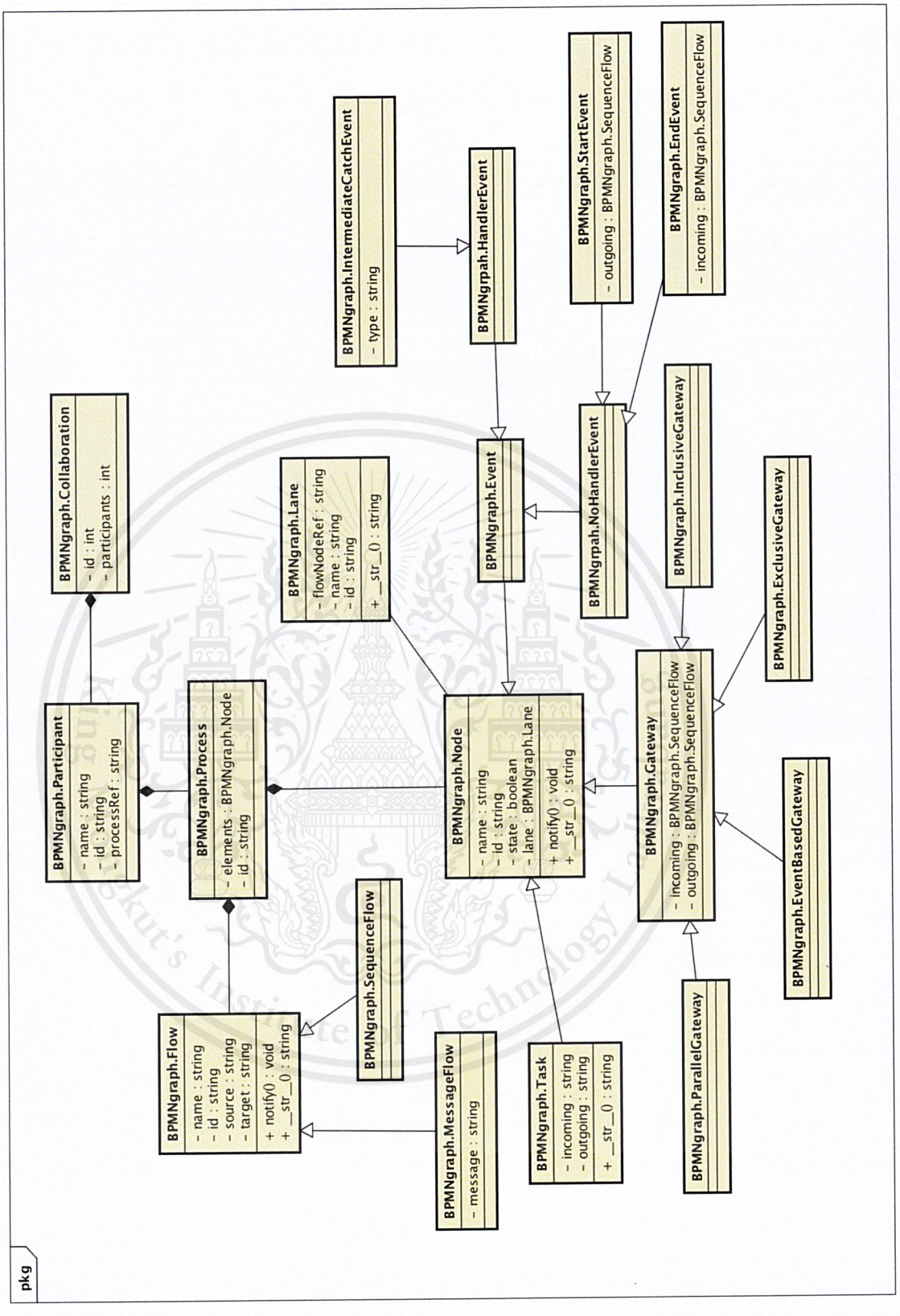


Figure 25: BPMN 2.0 Elements as an Object-based Class Diagram

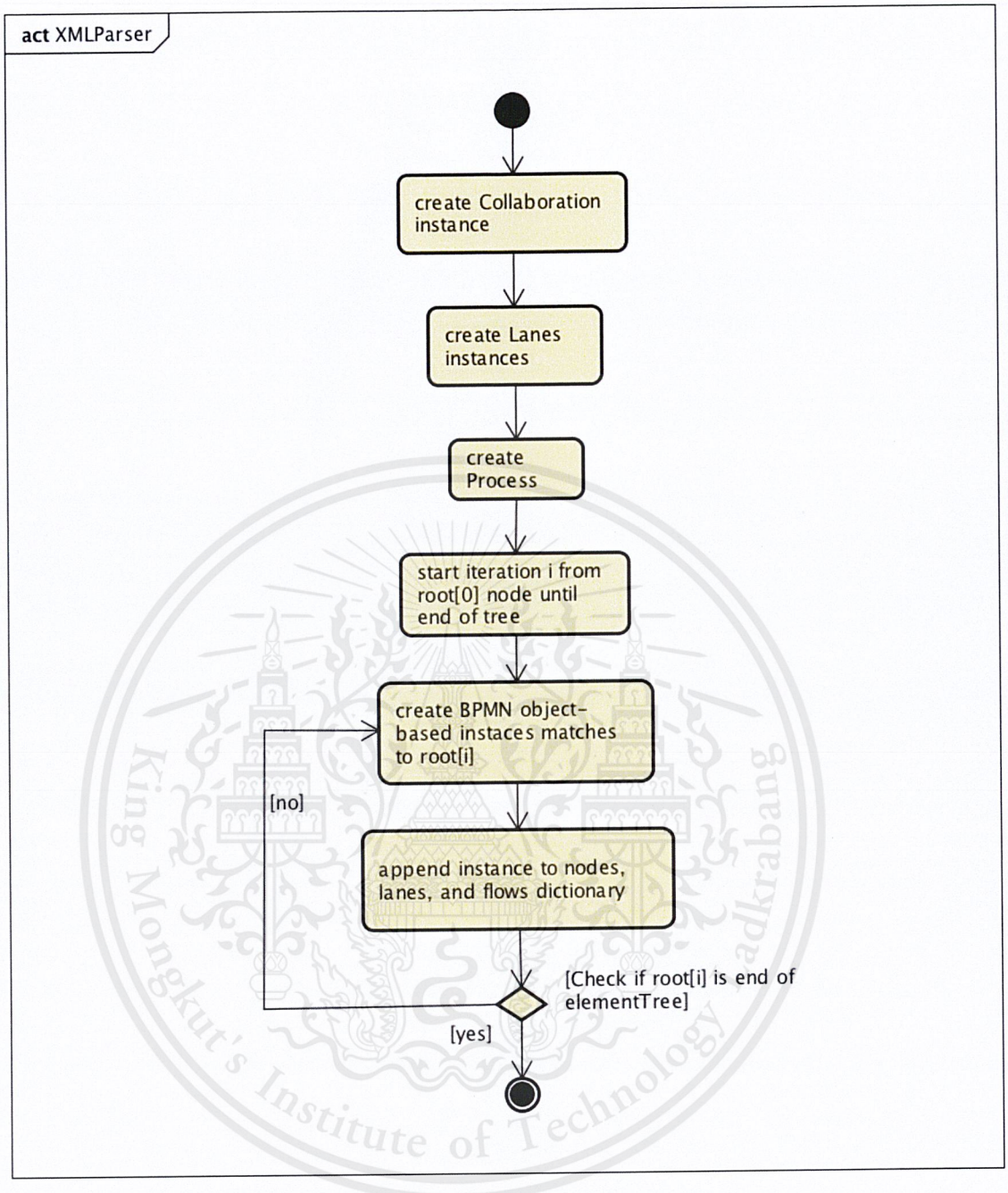


Figure 26: XML Parser Flow Chart

Concerning the XMLParser, its main function that handles all tasks of the element's mapping and collecting of processes is `createProcess` which first takes the root of the XML parse tree element as an only input variable, then iterates through the length of the parse tree, mapping an XML element to Python objects as described earlier. This method traverses into each root's child element to find its required attributes for creating an equivalent object-based instance such as its incoming flows,

outgoing flows, name, id, lane, or any documents that attached to the element then store the instance in the dictionaries variables of `processes`, `nodes`, and `flows`. Figure 24 shows the class diagram that depicts the XML parser's attributes and methods

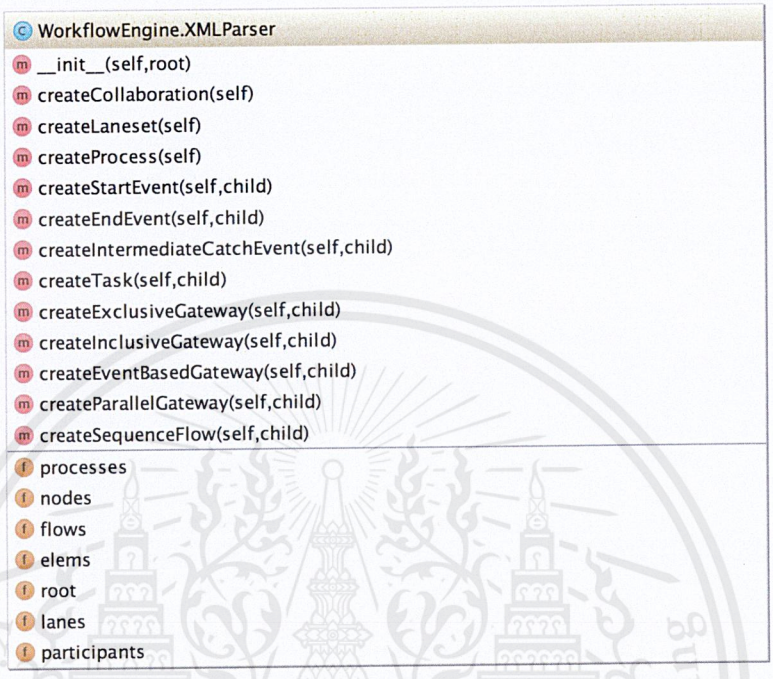


Figure 27: XML Parser's Attributes and Methods

6.3 Execution of Graph-based Workflows

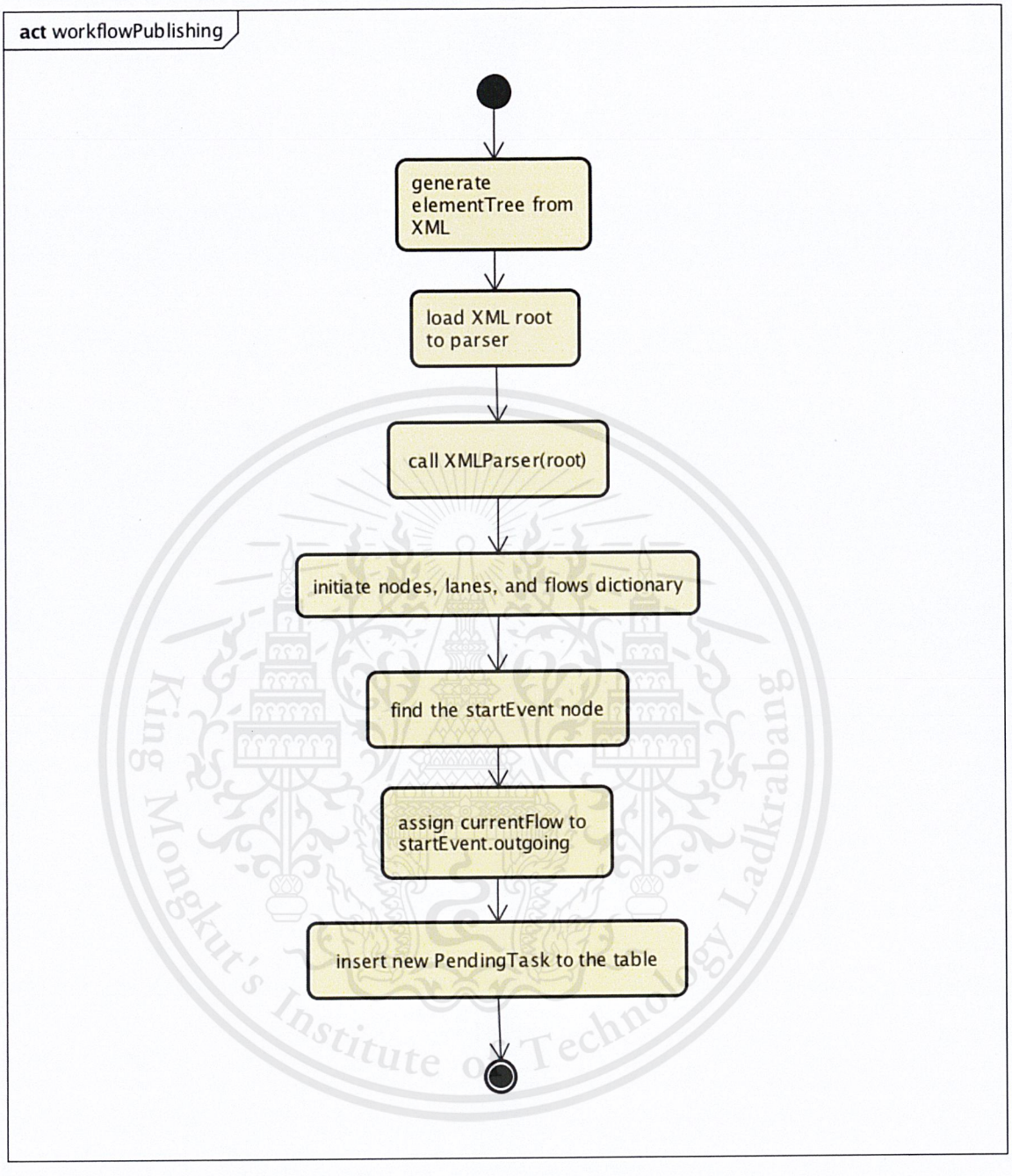


Figure 28: Execution of Workflow Template Flow Chart

During the process of traversing the flow graph, one of the most important and essential thing is to know where the flow is at in the graph. Consequently, we use one variable called `currentFlow` to handle this situation, this object is actually a `Flow` object instance consists of `source` and `target`of that flow. The execution of the graph-based workflow starts

after finding `StartEvent` instance is found in the dictionary storing all node elements (Activity) of the flows, then we set the `currentFlow` to be the one that outgoing of the `startEvent` node.

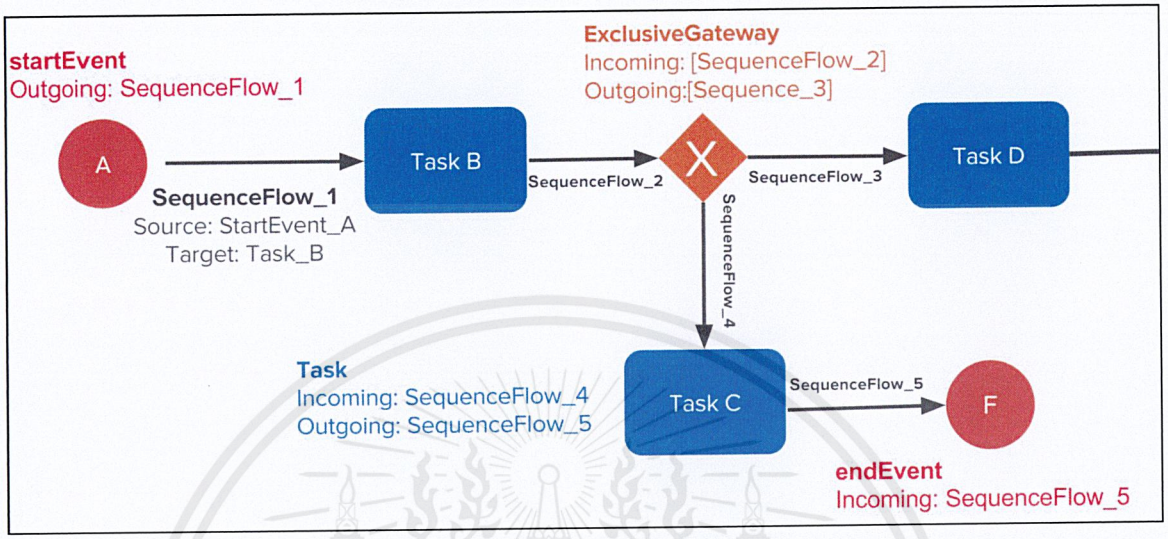


Figure 29: BPMN Flow Graph Representation

After we have set the `currentFlow`, the `source` of that `currentFlow` is set to be a source object and its state is updated to `Running` then we look further to current flow's target which is set to be the target object; to determine the type of this target object, we used python build-in function `isinstance` to execute that element the way as it should be running in real-life. For example, execution of an exclusive gateway, we need to collect all of its option decisions and its form to display for the assigned user then waiting for user to answer or executing their task, by doing this, we can determine the direction of the flow. As the result, the `currentFlow` is updated to the next flow.

In order to do all these steps, we need to have `PendingTask` as well as `ExecutingWorkflow` table to handle the temporary data created and maintained by the engine while a workflow is executing. After all the execution of the workflow is finished (reached the `EndEvent` element), this data will be released and then be deleted.

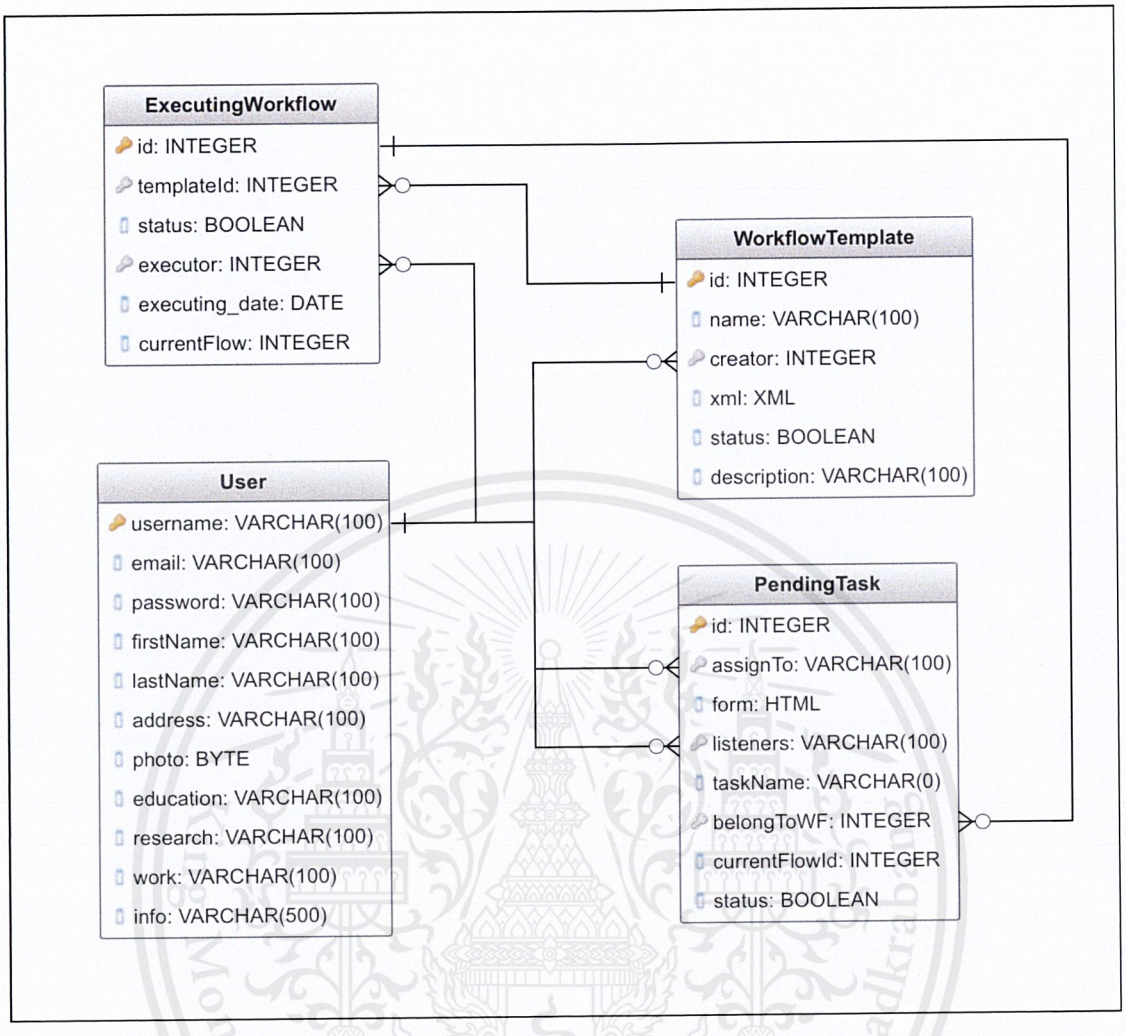


Figure 30: Workflow Engine Database Diagram of Tables Storing Temporary Data

When new task (node) is being executed, the new `PendingTask` instance is created and inserted into the table, `currentFlowId` is kept, `status` is setted to active, real-time notification for assigned user who is responsible of that task and the listeners of the task will be notified as well. If user finishes their task and submits the task execution, the engine has to update the status field of that task's row in Pending task table to inactive and update the `CurrentFlowId` to the target object (next object) then the process kept running repeatedly until it reaches the `EndEvent`.

6.3.1 Decision Making

After the engine reaching the target of the `currentFlow`, either that target is an Exclusive or an Inclusive Gateway, decision of which way to go must be determined by user's selection. All of the possible ways is shown as radio buttons for receiving user's input. The method of execution is explained below.

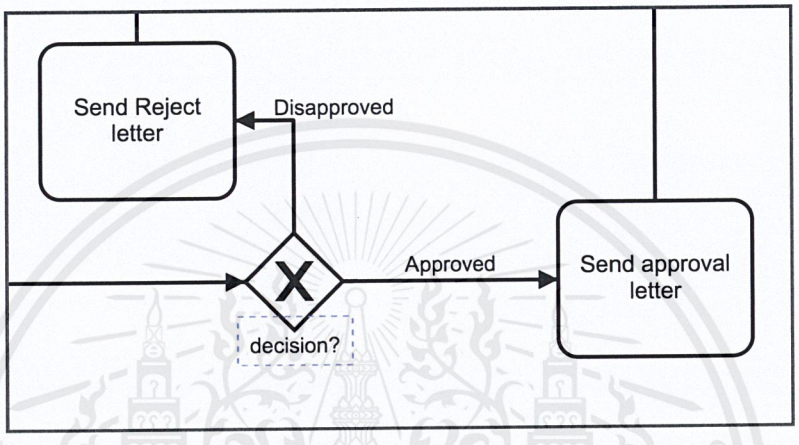


Figure 31: Exclusive Gateway with Two Outgoing Flows

The engine first creates the dictionary with name of each flow as a key and its id as a value as shown below, then returns this dictionary to `view.py`, the function then passes the dictionary to Django session for the next execution usage after it receives the user's input.

```
{
    'Disapproved': 'SequenceFlow_13djieg',
    'Approved':    'SequenceFlow_023mjn4'
}
```

Figure 32: Exclusive Gateway Decisions Represented in Python Dictionary

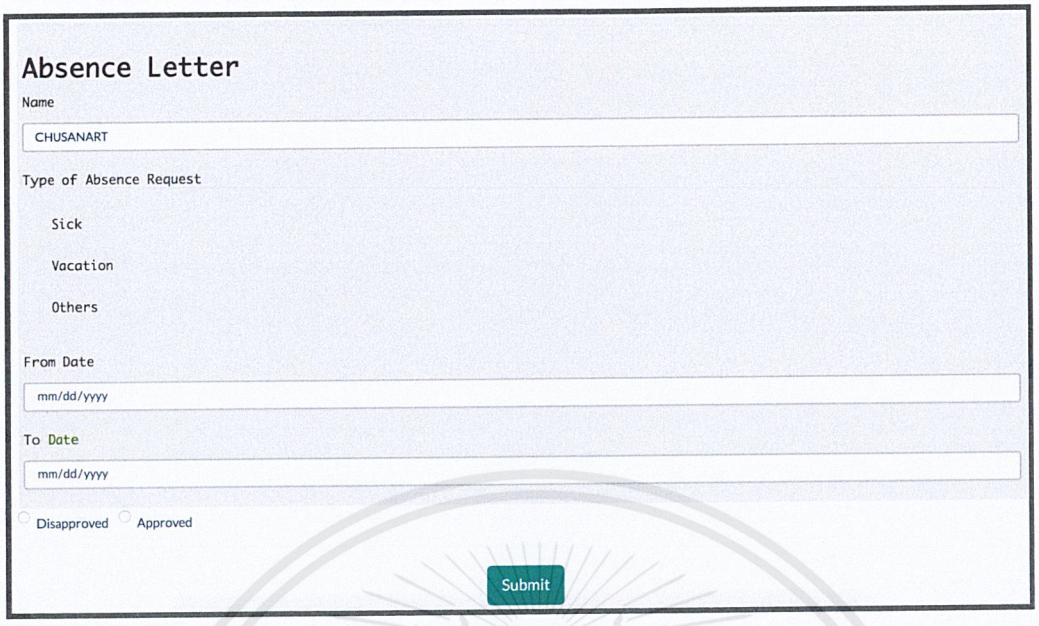


Figure 33: Exclusive Gateway Decisions Represented in User's Input Form

In order to map user's input with the flow id, we retrieve user's selection from the request and map it with the value in dictionary which we have passed through the session as a direction of executing next task by updating `currentFlow` to the id of user's decision flow.

6.3.2 Joining and Splitting of Parallel Gateway

The main important thing of executing the parallel gateway is to distinguish between Splitting and Joining of Parallel Gateway. For Splitting Parallel Gateway, there is only one incoming flow and multiple outgoing flow, we need to set all outgoing tasks to be active and assign to user who responsible for each task by create new PendingTasks rows into the database.

On the other hand, for Joining Parallel Gateway, we can detect multiple incoming flows with only one outgoing flow. At this point, we have to check all of the incoming tasks that if there exist a task that is still executing (except itself), the current incoming task will be delete and waits until there is no existing of any other incoming tasks left then the `currentFlow` can proceed to the next `sequenceFlow`.

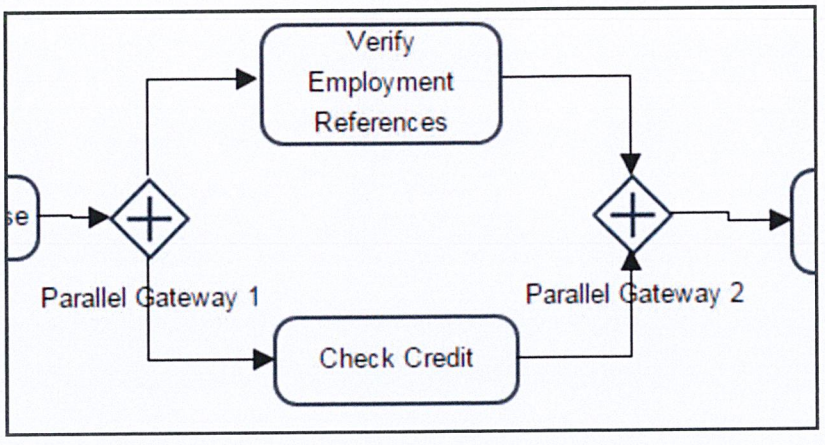


Figure 34: Splitting and Joining Parallel Gateway

6.3.3 Event Handling

There are two types of events that need to be handled, they are timers and messages.

The timer handler is used to stop the process of a given time or when activity timeouts that users are attached to it.

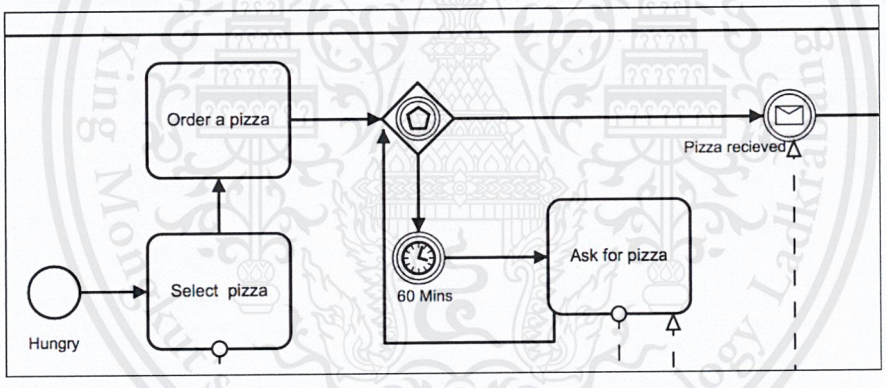


Figure 35: Example of Diagram with Event Handling

For a Timer; we added a property panel so that user can modify their diagram as they like and this feature also allow user to set timer in a task. This feature should resolve the time event handling by counting it down after the task has been started.

For a Message; with the sockets that we integrated in the project, this will allow the system to be able to know whether or not there is a message sending to the user

Chapter 7

Results and Evaluations

In this chapter, we demonstrate how the Workflow Engine is used with a case study.

7.1 Workflow Engine applied to a case study

A good case study for a real-world workflow is a online Pizza Delivery as it deploys many basic BPMN elements.

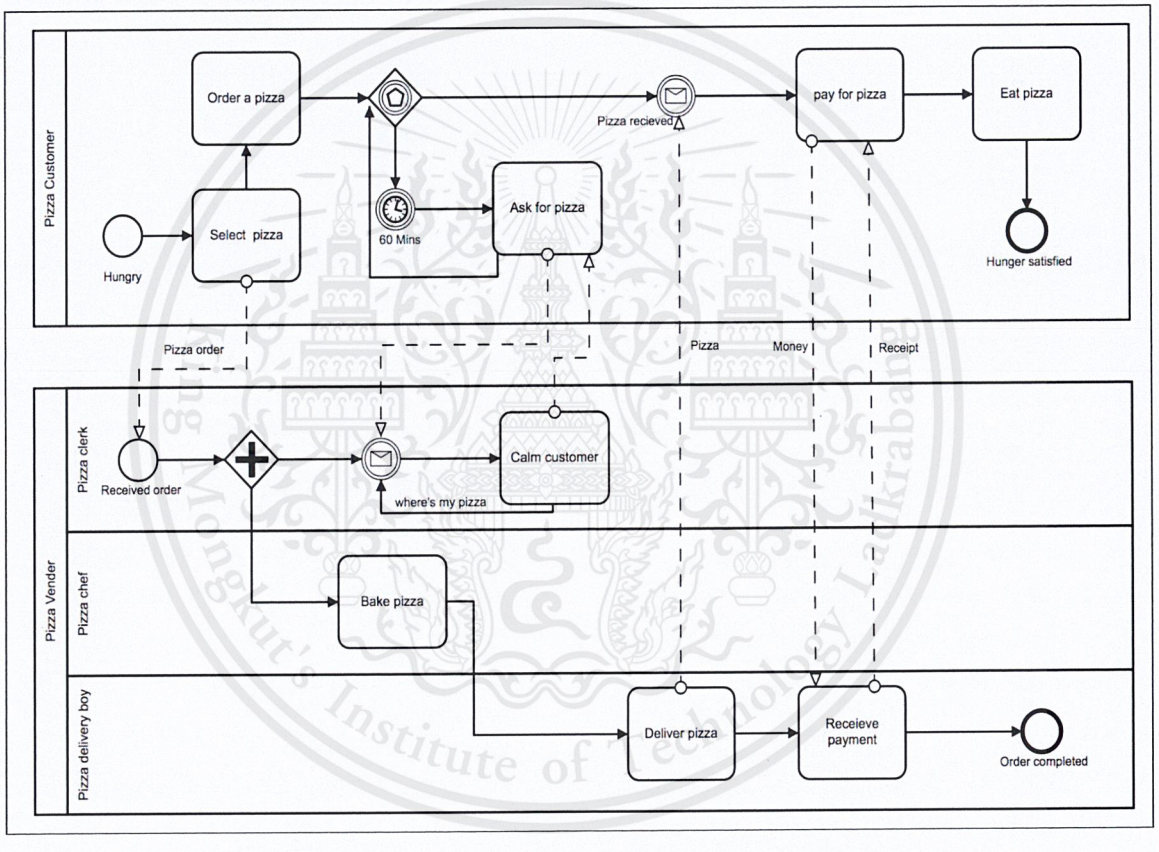


Figure 36: Pizza Delivery BPMN Diagram

As shown above, this diagram consists of two pools which are customer and pizza vendor. The process starts when the customer get hungry, he then uses an online pizza website to order a pizza. He chooses his pizza and its topping then submits his order. As you can see the dash line or the message flow, the pizza vendor now receives the order as both lane are active simultaneously. After he enters the waiting state at the event-based gateway, he is now waiting for his pizza. If the order takes longer than sixty minutes the customer can call the pizza shop and ask about his pizza.

The pizza vendor is now entered the parallel gateway by having the chef bake the pizza while the clerk is waiting for the customer call. After the chef gives out his final preparation of the pizza. The delivery boy will deliver the pizza to the customer. When the customer received his pizza, he will exit the waiting state at the event-based gateway and proceed to pay for his pizza. Once he gives the money to the delivery boy and the boy returns him with a receipt. The final task for him before ending the workflow is enjoying eating his pizza.

If unfortunately the alternative event happens when the chef accidentally put the wrong topping on his pizza, twice, resulting in the pizza couldn't make it to the customer house within sixty minutes. The customer is then calling and asking for his pizza. The clerk will calm the customer down as the customer continue waiting for his pizza.

7.2 Demonstration (Screen Captures)

Figure 37: User index page

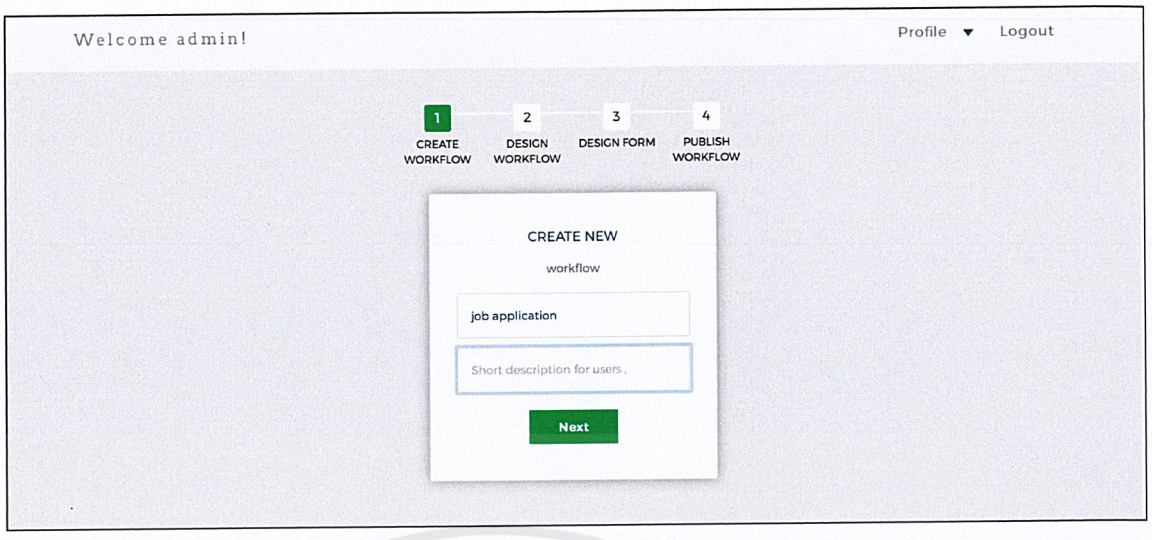


Figure 38: Create new workflow page

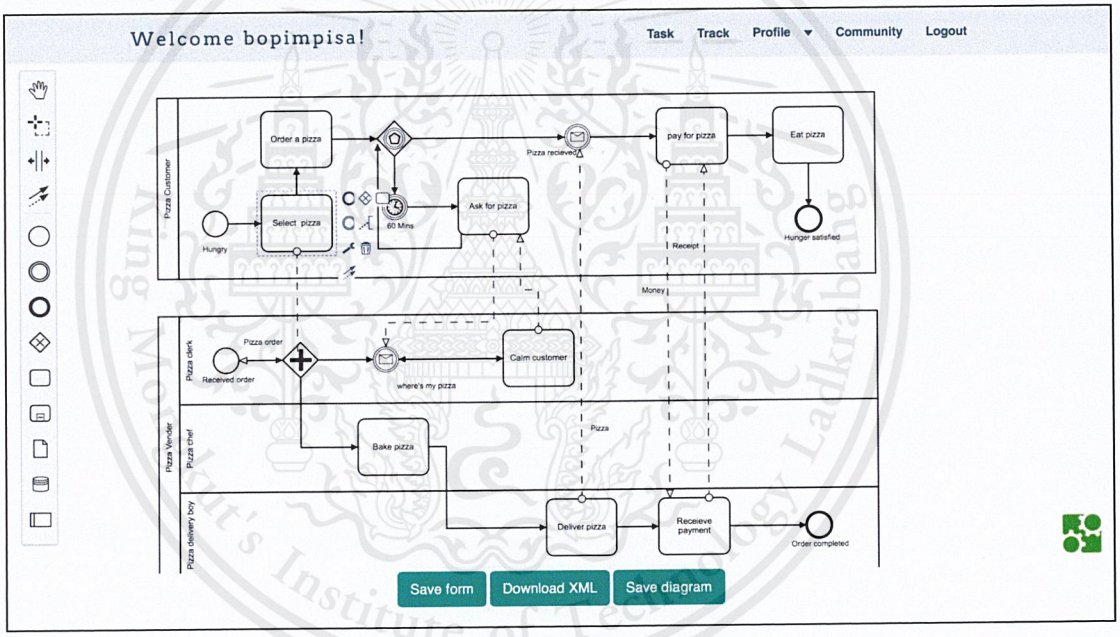


Figure 39: BPMN drawing tool

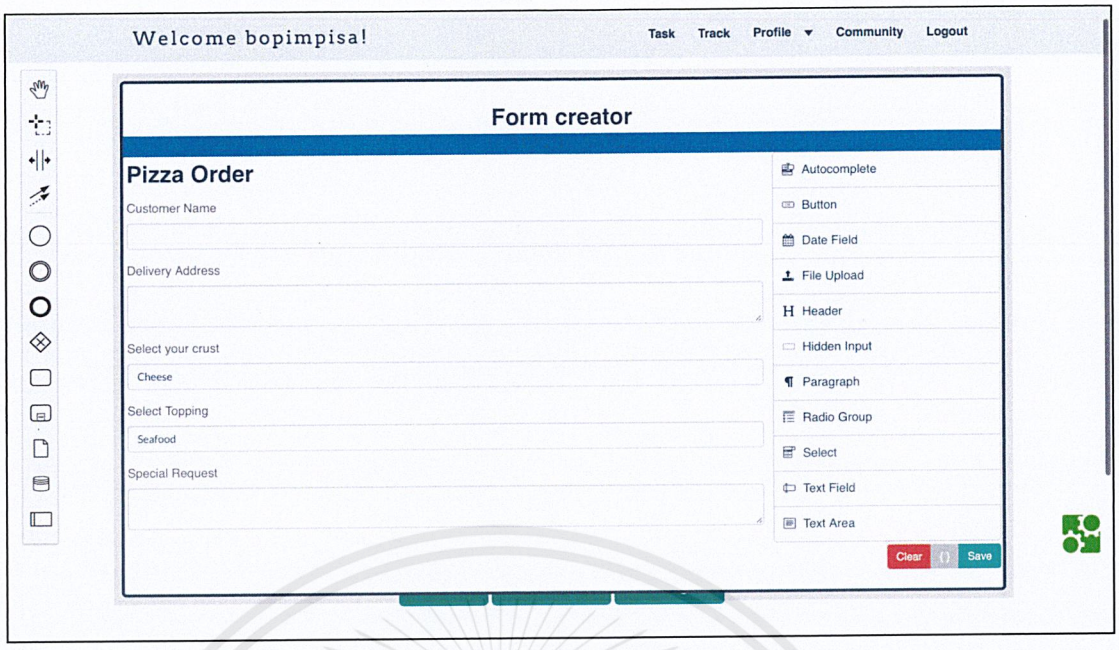


Figure 40: Form creator(1)

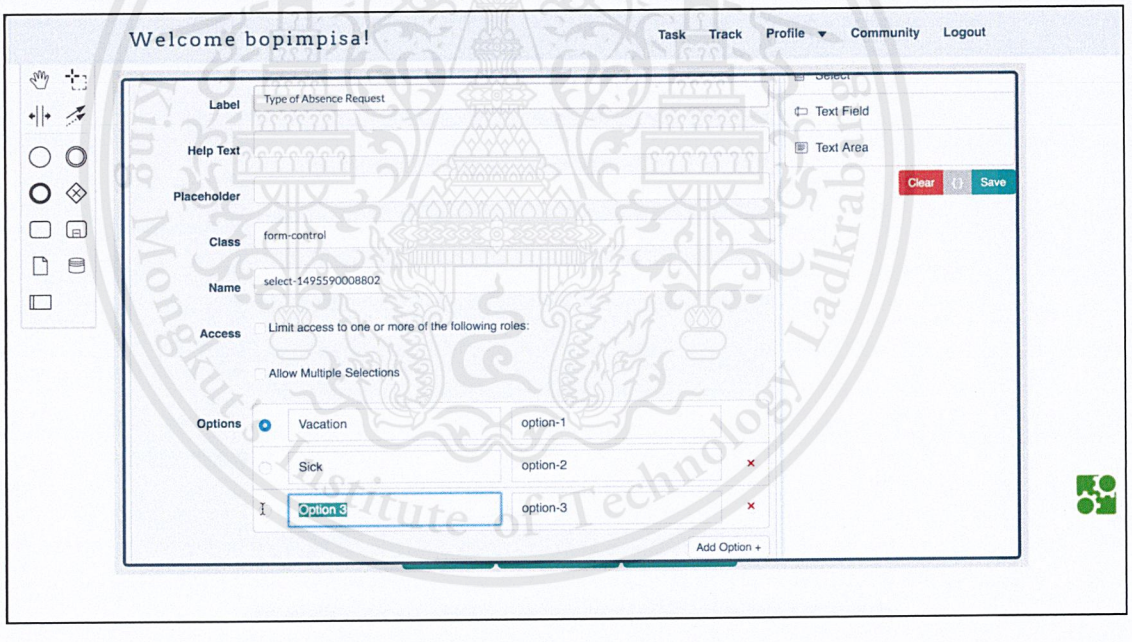


Figure 41: Form creator(2)

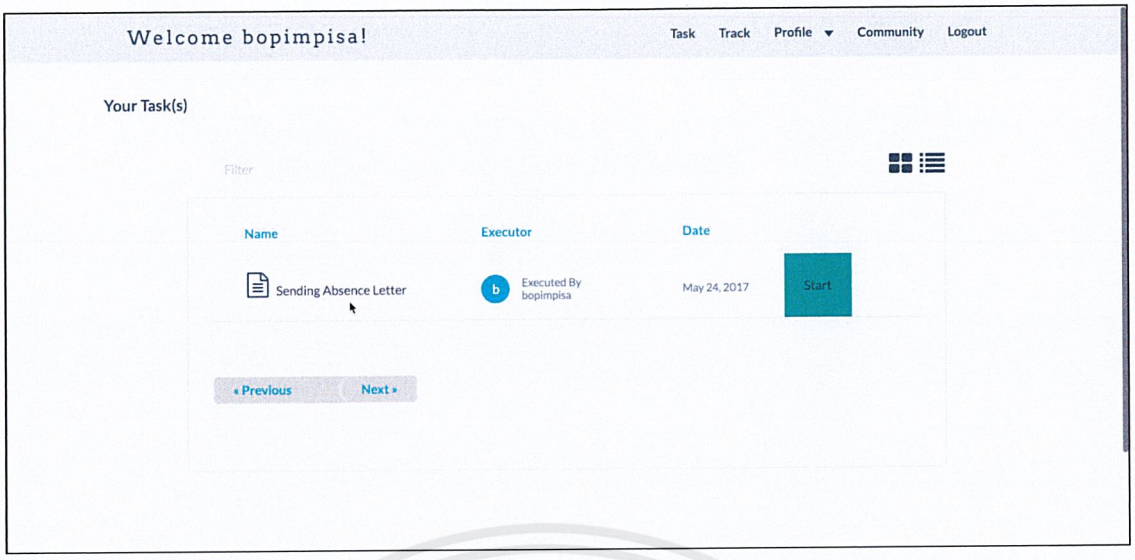


Figure 42: Task notification

Figure 43: Executing User Task (Fill in form)

Figure 44: Executing User Task (Exclusive Gateway)

Figure 45: Executing User Task (Task Submission)

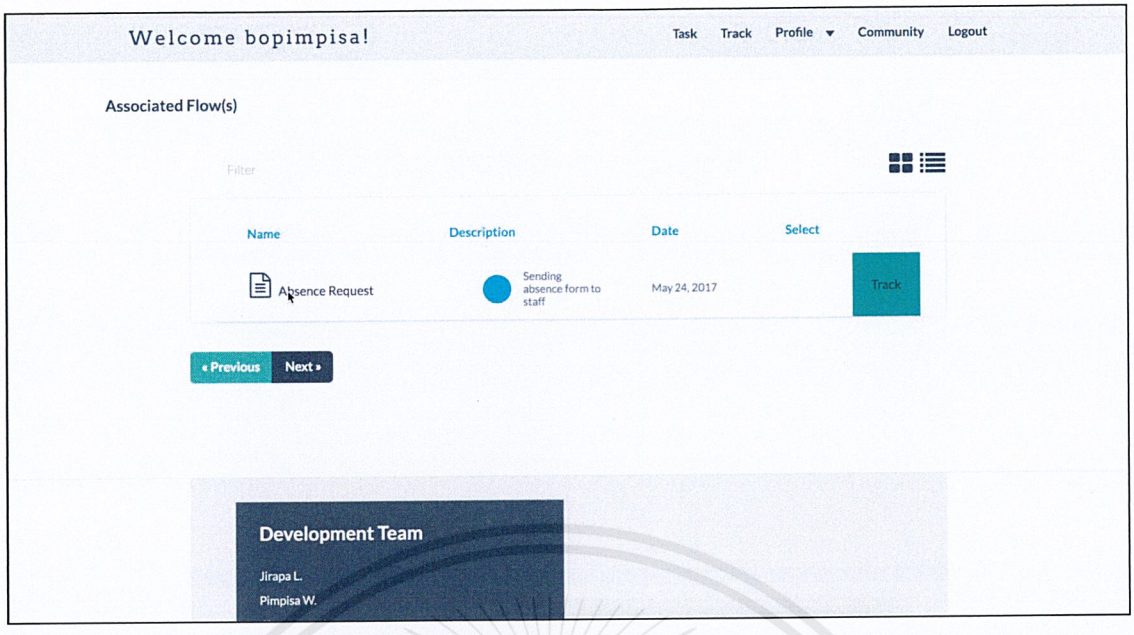


Figure 46: Track User's Executing Flow

7.3 Evaluations

Although the Workflow Engine we have developed in this project is still in the prototype stage, we try to test it with many real-world example where most of BPMN basic elements are used in the workflows. With further careful analysis to full range of the BPMN language we can ensure that this system can be released for the public.

Chapter 8

Conclusion

The Workflow Engine we have created in this project is a novel idea, there is still room for other ideas to apply to it.

8.1 Future Works

There are other aspects of the Workflow Engine which can be entered, including

- Real time collaborations between users, so that they can design and draw diagram at the same time. Similarly to Google docs and slides.
- Administrator will be hundred percent in charge of approving all the to-be-publish workflows beforehand.
- Connecting workflow with others, building network of workflows. And connecting service with service then turn workflow into script for reusable purposes.
- Implementing a tool for drawing GUI. We are now provides the tool for drawing BPMN diagram, so in the future itimemachine.net should be able to draw even more GUI such as a graph.

Appendix

1.) BPMN process elements

The Process Modeling Conformance type set consists of three subclasses as follows

- Descriptive Conformance Subclass
- Analytic Conformance Subclass
- Common Executable Conformance Subclass

Descriptive Conformance Sub-Class

The Descriptive Conformance Subclass elements are shown in Table A.

Table A – Descriptive Conformance Subclass Elements and Attributes

Element	**Attributes**
participant	id, name, processRef
laneSet	id, lane with name, childLaneSet, flowElementRef
sequenceFlow (unconditional)	id, name, sourceRef, targetRef
messageFlow	id, name, sourceRef, targetRef
exclusiveGateway	id, name
parallelGateway	id, name
task (None)	id, name
userTask	id, name
serviceTask	id, name
sExpandedSubProcess	flowElement
callActivity	id, name, flowElement
dataObject	id, name

textAnnotation	id, text
association	id, name, sourceRef, targetRef, associationDirection
dataStoreReference	id, name, dataStoreRef
startEvent (None)	id, name
endEvent (None)	id, name
messageStartEvent	id, name, messageEventDefinition
messageEndEvent	id, name, messageEventDefinition
timerStartEvent	id, name, timerEventDefinition
terminateEndEvent	id, name, terminateEventDefinition
Documentation	text
Group	id, categoryValueRef

Analytic Conformance Subclass

The Analytic Conformance Subclass contains all the elements of the Descriptive Conformance Subclass(Table A and the elements as shown in Table B.

Table B - The Analytic Conformance Subclass Elements and Attributes

Element	**Attributes**
sequenceFlow (conditional)	id, name, sourceRef, targetRef, conditionExpressiona
sequenceFlow (default)	id, name, sourceRef, targetRef, default
sendTask	id, name
receiveTask	id, name
Looping Activity	standardLoopCharacteristics
MultiInstance Activity	multiInstanceLoopCharacteristics

exclusiveGateway	default attribute
inclusiveGateway	id, name, eventGatewayType
eventBasedGateway	id, name, eventGatewayType
Link catch/throw Intermediate Event	Id, name, linkEventDefinition
signalStartEvent	id, name, signalEventDefinition
signalEndEvent	id, name, signalEventDefinition
Catching message Intermediate Event	id, name, messageEventDefinition
Throwing message Intermediate Event	id, name, messageEventDefinition
Boundary message Intermediate Event	id, name, attachedToRef, messageEventDefinition
Non-interrupting Boundary message Intermediate Event	id, name, attachedToRef, cancelActivity=false, messageEventDefinition
Catching timer Intermediate Event	id, name, timerEventDefinition
Boundary timer Intermediate Event	id, name, attachedToRef, timerEventDefinition
Non-interrupting Boundary timer Intermediate Event	id, name, attachedToRef, cancelActivity=false, timerEventDefinition
Boundary error Intermediate Event	id, name, attachedToRef, errorEventDefinition
errorEndEvent	id, name, errorEventDefinition
Non-interrupting Boundary escalation Intermediate Event	id, name, attachedToRef, cancelActivity=false, escalationEventDefinition
Throwing escalation Intermediate Event	id, name, escalationEventDefinition
escalationEndEvent	id, name, escalationEventDefinition
Catching signal Intermediate Event	id, name, signalEventDefinition

Throwing signal Intermediate Event	id, name, signalEventDefinition
Boundary signal Intermediate Event	id, name, attachedToRef, signalEventDefinition
interrupting Boundary signal Intermediate Event	id, name, attachedToRef, cancelActivity=false, signalEventDefinition conditionalStartEvent
conditionalStartEvent	id, name, conditionalEventDefinition
Catching conditional Intermediate Event	id, name, conditionalEventDefinition
Boundary conditional Intermediate Event	id, name, conditionalEventDefinition
Non-interrupting Boundary condi- tional Intermediate Event	id, name, cancelActivity=false, conditionalEventDefinition
message	id, name, add messageRef attribute to messageFlow

Common Executable Conformance Subclass

The Common Executable Conformance Subclass contains all the elements in The Descriptive Conformance Subclass(Table A and in The Analytic Conformance Subclass (Table B and its supporting classes in Table C and Table D, which listed below.

Table C - The Common Executable Process Modeling Conformance Subclass

Element	**Attributes**
sequenceFlow (unconditional)	id, name, sourceRef, targetRef
sequenceFlow (conditional)	id, name, sourceRef, targetRef, conditionExpression
sequenceFlow (default)	id, name, sourceRef, targetRef, default
isExpandedSubProcess	id, name, flowElement, loopCharacteristics, boundaryEventRefs

exclusiveGateway	id, name, gatewayDirection (only converging and diverging), default
parallelGateway	id, name, gatewayDirection (only converging and diverging)
startEvent (None)	id, name
endEvent (None)	id, name
eventBasedGateway	id, name, gatewayDirection, eventGatewayType
userTask	id, name, renderings, implementation, resources, ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics, boundaryEventRefs
serviceTask	id, name, implementation, operationRef, ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics, boundaryEventRefs
callActivity	id, name, calledElement, ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics, boundaryEventRefs
dataObject	id, name, isCollection, itemSubjectRef
textAnnotation	id, text
dataAssociation	id, name, sourceRef, targetRef, assignment
messageStartEvent	id, name, messageEventDefinition (either ref or contained), dataOutput, dataOutputAssociations
messageEndEvent	id, name, messageEventDefinition, (either ref or contained), dataInput, dataInputAssociations
terminateEndEvent	Id, name , terminate EventDefinition
Catching message Intermediate Event	id, name, messageEventDefinition (either ref or contained), dataOutput, dataOutputAssociations

Throwing message Intermediate Event	id, name, messageEventDefinition (either ref or contained), dataInput, dataInputAssociations
Catching timer Intermediate Event	id, name, timerEventDefinition
Boundary error Intermediate Event	id, name, attachedToRef, errorEventDefinition, dataOutput, dataOutputAssociations

Table D - The Common Executable Subclass Supporting Elements.

Element	**Attributes**
StandardLoopCharacteristics	id, loopCondition
MultiInstanceLoopCharacteristics	id, isSequential, loopDataInput, inputDataItem
Rendering	
Resource	id, name
ResourceRole	id, resourceRef, resourceAssignmentExpression
InputOutputSpecification	id, dataInputs, dataOutputs
DataInput	id, name, isCollection, itemSubjectRef
DataOutput	id, name, isCollection, itemSubjectRef
ItemDefinition	id, structure or import
Operation	id, name, inMessageRef, outMessageRef, errorRefs
Message	id, name, structureRef
Error	id, structureRef
Assignment	id, from, to (defined by an XSD Complex Type)
MessageEventDefinition	id, messageRef, operationRef
TerminateEventDefinition	id
TimerEventDefinition	id, timeDate

References

[1] - Microsoft (2016) Microsoft Flow [online]
Available: (https://flow.microsoft.com/en-us)

[2] - IFTTT Inc (2016) Learn how IFTTT works - IFTTT [Online]
Available: (https://ifttt.com)

[3] - Camunda BPM (2016) Platform For Workflow and Business Process [Online]
Available: (https://camunda.org)

[4] - Alfresco (2016) Activate Process and Content to Make Business Flow [Online]
Available: (https://www.alfresco.com)

[5] - Object Management Group (2016) Business Process Model and Notation [Online]
Available: (https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation)

[6] - Introduction to BPMN by Stephen A. White, IBM Corporation, 2014 Oct.

[7] - Business Process Model and Notation, v2.0, OMG Doc Frml/11-01-03