WEB-BASED WORKFLOW MANAGEMENT SYSTEM

JARATRAWEE DEEKHUEANKHAN
KANANEK ATICHATVIWAT
KIATTISAK KESORNBUA

BACHELOR OF ENGINEERING PROGRAM IN SOFTWARE ENGINEERING
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2015

Thesis – Academic Year 2015

B.Eng. in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: Web-based Workflow Management System

Authors:

Mr. Kananek Atichatviwat	Student ID: 55090003
Mr. Kiattisak Kesornbua	Student ID: 55090008
Mr. Jaratrawee Deekhueankhan	Student ID: 55090009

Approved for submission

.....

(Dr. Natthapong Jungteerapanich)

Advisor

Date 31/5/2016

.....

(Asst.Prof.Dr. Chaiwat Nuthong)

Co-advisor

Date 31/5/2016

Web-based Workflow Management System

Mr. Kananek Atichatviwat 55090003
Mr. Kiattisak Kesornbua 55090008
Mr. Jaratrawee Deekhueankhan 55090009
Dr. Natthapong Jungteerapanich Advisor
Asst.Prof.Dr. Chaiwat Nuthong Co-Advisor
Academic Year 2015

ABSTRACT

Good management of workflows within an organization is crucial to the efficiency of the operation of the organization. Many organizations have started to use software applications to help manage their workflows. There are a number of the so-called workflow management software applications in the market. The commercial options tend to be expensive, while the free and/or open-source ones are either hard to use or not powerful enough to capture complicated real-world workflows. In this project, we have developed a web-based workflow management software application which supports BPMN 2.0 notation, an industry-standard process modelling notation, while being easy to use. The user can design and write their workflows in BPMN notation. The workflow designed can then be executed and used by members of the organization through web browsers. The application was developed primarily using JavaScript and utilized a number of modern open-source libraries and software tools, including Node.js, Express web framework, Mongoose ODM, jQuery and Bootstrap. Our project is part of a larger project called "Monkey Office", which aims at developing web-based applications to support the internal management of the International College, KMITL.

Acknowledgements

The success of the Web-based Workflow Management System is a reflection of the efforts and support of numerous people and organizations to all of whom we are grateful. The project will not be possible without the help of so many people in so many ways.

We would like to express the deepest appreciation to our advisor, Dr. Natthapong Jungteerapanich, who has the attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in regard to research, and excitement in regard to teaching. Without his guidance and persistent help, this project will not be possible. In addition, we would like to thank our co-advisor, Asst.Prof.Dr. Chaiwat Nuthong for the contribution, suggestion, and encouragement towards the completion of this project. Futhermore our appreciation also extends to Dr. Isara Anantavrasilp who gives us the knowledgement about developing this project.

Finally, we would like to show gratitute to our friends and all who contributed in one way or the other in the course of this project. Especially, the group whose work is to develop MonkeyOffice. We also really appreciate a chance to have a collaboration with other groups as well.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter provides an explanation on the motivation, the objectives, the contributions, and the scope of work of this project, as well as the structure of this thesis. Section 1.1 describes the motivation. Section 1.2 shows the main objectives of the project. Section 1.3 provides the scope of work. Finally, section 1.4 describes the organization of this thesis.

1.1 Motivation

Everything that we do in our daily life can be seen as a process, e.g. planning holidays, acquiring goods, preparing a meal, writing an email, etc. In business situations, a business process is an activity or set of activities performed by one or more business participants in order to deliver the final product or service to its customers. For example, take the business of a mortgage loan company in which a customer applies for a loan. This business process involves activities related to checking the credibility of the customer, getting his information, calculating his monthly fee, and many more. All these activities are linked to each other, so that they occur at the right time and order. The input to this business process is the customer's information and the output is a signed loan contract. Business processes can be categorized into three types [2], as follows: management processes, operational processes, and supporting processes. Management processes govern the operation of a particular organization's system of operations. Operational processes constitute the core business. Supporting processes, such as human resources and accounting, are put in place to support the core processes [3]. When we are involved in process modeling for a business, are considered the management, operation and support aspects of the business in question.

A business process is sometimes referred to as a workflow. These two terms are very similar and can often be used interchangeably. Unfortunately, there are no standard definitions for these terms. One common definition is that a workflow is a sequence of activities through which a piece of work passes from initiation to completion, while a business process is a workflow that serves or affects a business goal. For example, placing orders or replenishing the stock are examples of business processes. In contrast, the workflows for requesting a sick leave or destroying document are not typically considered business processes.

This project is a part of a larger system, named "MonkeyOffice", which is a web-based software application that aims to facilitate the management of documents and workflows at the International College, KMITL. At present, the operations in the organization are mostly manual. The college staff has a word processor that assists them in writing or editing documents. The number of documents increases everyday. With a large number of documents, this has led to the problem of storing and searching the documents. For this reason, the International College would like to have a system that helps storing and searching for documents. Thus, the document management system comes into action. Additionally, the documents have to flow through many departments during the workflow. Very often, documents get lost or forgotten along the way. To eliminate this problem, the organization has asked for a computerized system which can help track the documents as they follow a workflow or, even better, a system which could help automate entire workflows. Presently, there are many similar systems available in the market, but the main problem is that the budget for purchasing software is limited and the costs of good workflow management software are typically very high. Moreover, existing workflow management software requires programming skills and a steep learning curve. This project has thus been initiated to develop a workflow management software application that can handle the complex workflows at the International College while being easy to learn.

1.2 Objectives

The web-based workflow management system is developed to support the International College, KMITL. There are two main objectives as follows:

1. The production of documents within the International College becomes more automated, more efficient, and less error-prone.
2. The management of workflows within the organization are computerized and can be tracked.

These objectives provide many benefits to the organization such as giving results in fewer administrative errors as a result of it can be used to ensure that processes are undertaken with the steps in the right order. Additionally, the system also helps the various departments stay in touch with one another by simplifying the channels through which these entities communicate, and many more.

1.3 Scope of work

1. Study the concepts of business process modeling and workflow managment; study the details of BPMN 2.0, both from the user's point of view and the developer's point of view.
2. Survey existing workflow management software.
3. Develop a web-based software application that allows the user to create a workflow description using BPMN 2.0 notation, automate the execution of a workflow, and manage the workflow executions. The application to be developed is to be a part of the Monkey Office software system and must work with other parts of the system.
4. Test the developed web application on selected actual workflows in the International College.
5. Deploy the developed web application as part of the Monkey Office system at the International College.

1.4 Thesis structure

The rest of this thesis is organized as follows:

- Chapter 2 provides a survey of related work. A comparative review of popular workflow management systems, namely, KiSSFLOW, Bonita BPM, and Activiti, is given.
- Chapter 3 provides a brief introduction to BPMN 2.0.
- Chapter 4 discusses requirements specification including functional requirements, non-functional requirements, and use case diagram and its description.
- Chapter 5 gives a description on software design including the overall architecture of the system, user role management, data management and workflow execution respectively.
- Chapter 6 gives an explanation on the development techniques, including Node.js, MongoDB, and Promise.
- Chapter 7 shows the result of developing this project.
- Chapter 8 gives a conclusion.

Chapter 2
Related Work

This chapter is a survey of some existing workflow management systems in the market, namely, KiSSFLOW, Bonita BPM, and Activiti. Sections 2.1 – 2.3 provide an overview of these three systems. Section 2.4 contains a comparative review of these systems.

2.1 KiSSFLOW

KiSSFLOW is a cloud-based workflow management software developed by OrangeScape. It was launched in Google I/O 2012 and is being used today by over 10,000 customers around the world. KiSSFLOW allows users to create a workflow in five easy steps without any coding.

There are six key features of KiSSFLOW. The first one is called Smart Workflows. A Workflows tend to become complex when the user implements exception flows such as rejection and errors. To keep the workflow simple, KiSSFLOW enables the user to define the normal flow without defining the exception flows. The second feature, called Easy 5-Step Process Creation Wizard, which guides the user through the process of workflow creation. With this feature, even the user with little or no technical knowledge can easily build their workflow in minutes. The third feature is Google Apps integration. KiSSFLOW can work with Google Docs, Contacts, Mail and Login (SSO) to provide a seamless Google experience. The fourth feature is Google cloud infrastructure. KiSSFLOW runs on reliable and scalable Google infrastructure (Google App Engine and Google Cloud SQL). As a result, the user can get the benefit of control and security along with the power of Google's self-managed cloud. The fifth one is organizational hierarchy support. KiSSFLOW allows the administrator to define the user hierarchy/reporting structure for the active users on KiSSFLOW. Lastly, KiSSFLOW comes with pre-built reports and a dashboard that can be used to view, analyze and gather insights about the workflows.

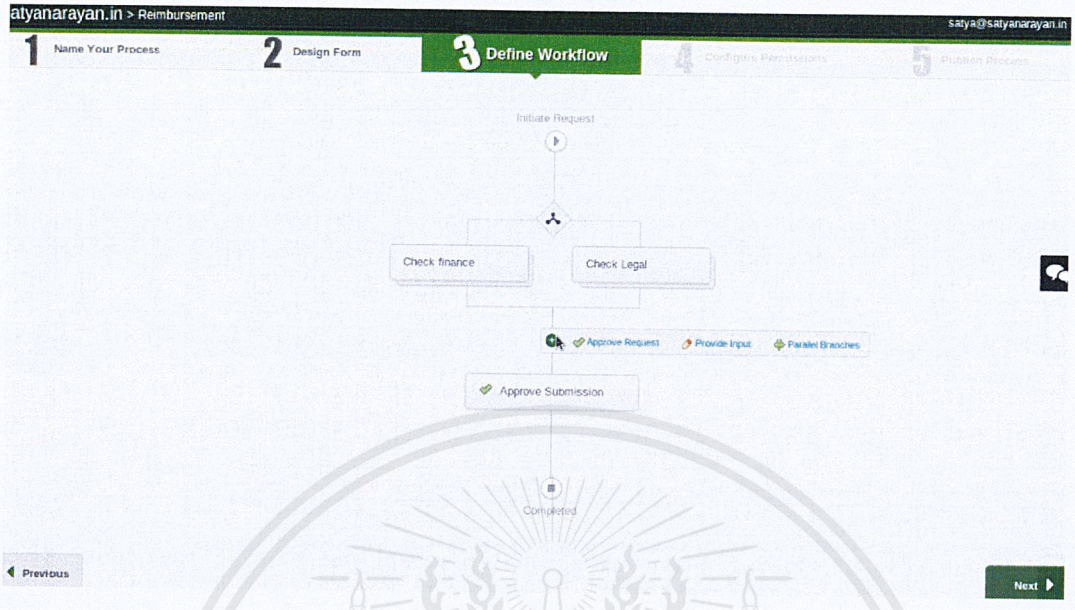

Figure 2.1. KiSSFLOW

2.2 Bonita BPM

Bonita BPM [8] is an open-source workflow management system developed by BonitaSoft Inc. Bonita BPM supports Groovy [9], an object-oriented scripting language for the Java platform. It is dynamically compiled to Java byte code and interoperates with other Java code and libraries [10].

There are three major components of Bonita BPM: Bonita Studio, Bonita BPM Engine, and Bonita Portal [10]. Bonita Studio allows the user to graphically create and modify business processes using BPMN 2.0 notation and connect those processes to other information systems within the organization to generate an autonomous web-based application. The second major component is Bonita BPM Engine, which is a JAVA API that allows the developer to interact programmatically with their processes. This requires JAVA programming skills. Finally, Bonita Portal is a portal website that allows each end-user to manage their processes in a webmail-like interface.

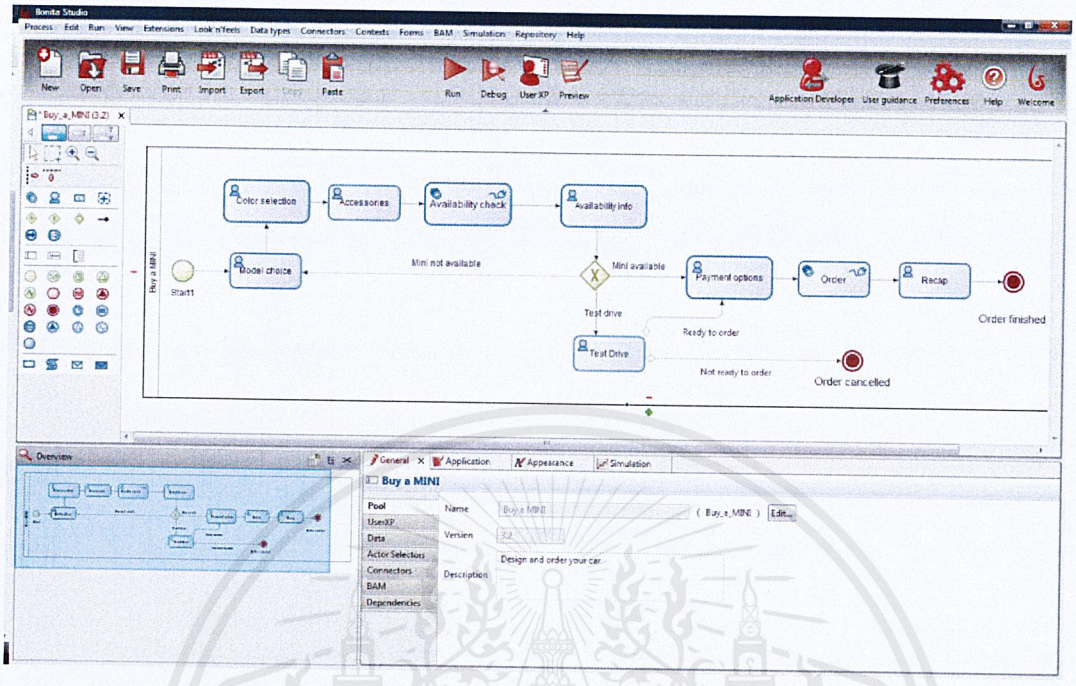

Figure 2.2. *Bonita BPM*

2.3 Activiti

Activiti [14] is a BPMN 2.0 process engine framework developed by the same team behind Alfresco, a famous document management system. Activiti is an independent open-source project, separate from Alfresco, but for some kinds of functionality, Activiti can be integrated into the Alfresco system to provide the necessary workflow engine capabilities, such as review and approval process for documents.

There are five components in the Activiti tool stack: Activiti Engine, Activiti Modeler, Activiti Designer, Activiti Explorer, and Activiti REST. The core component of the Activiti framework is the Activiti Engine that performs the process engine functions, such as executing BPMN 2.0 business processes and creating workflow tasks. Both of business and information analysts are capable of modeling a BPMN 2.0-compliant business process in a web browser as the result of the characteristics of the Activiti Modeler, which means that business processes can be shared easily. The Activiti Designer is an Eclipse-based plugin, which enables a developer to enhance the modeled business process into a BPMN 2.0 process that can be executed on the Activiti process engine. The Activiti Explore is a web tool to deploy process definitions, start new process instances and carry-out on workflows. Finally, the Activiti REST component, which provides a web application that starts the Activiti process engine when the web application is started. In addition, it offers a REST API that enables you to communicate

remotely with the Activiti Engine.

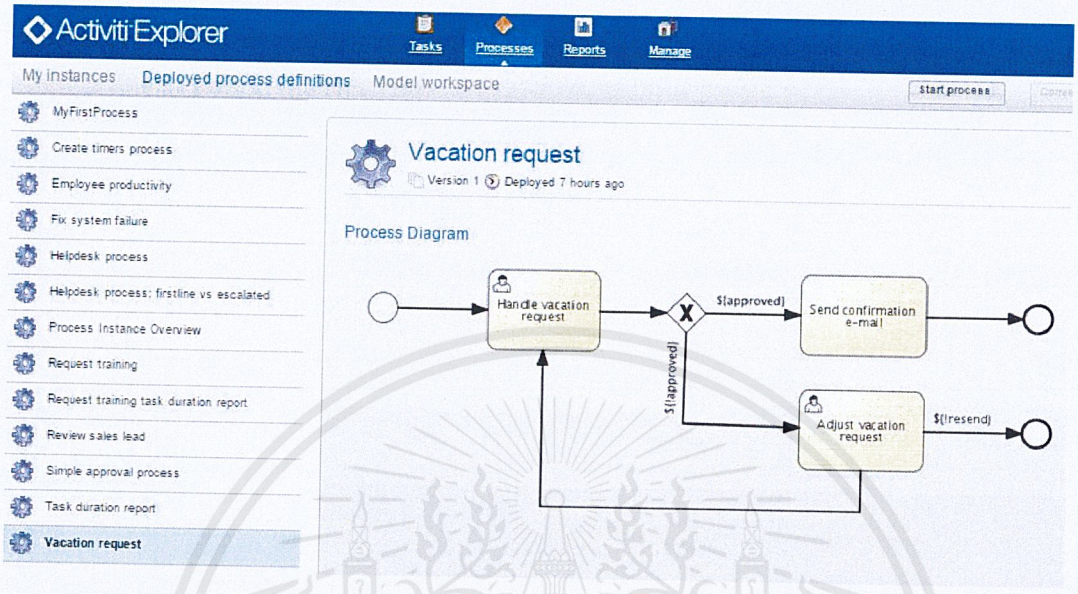

Figure 2.3. *Activiti explorer*

2.4 Discussions

The disadvantage of using KiSSFLOW is that it cannot be used to create a complex workflow and it did not support BPMN 2.0. While Bonita BPM is developed to be used for handling a complex workflow, but organizations that are considering to use Bonita BPM may have to pinpoint an expert in conducting their workflow processes as it requires some programming skill from users. Finally, Activiti is very similar to Bonita BPM, but it uses Unified Expression Language (UEL) to support expressions that may be used in BPMN. These expressions can be used to resolve and compare primitives, beans, list, arrays and maps. Bonita BPM supports a similar feature using Groovy scripting language.

Chapter 3
Business Process Modeling and Notation

This chapter provides an explanation on BPMN 2.0 and also gives you some practical examples about how to use BPMN 2.0 notation to model business processes. There are many different kinds of BPMN 2.0 elements, but we select only the ones that are commonly used in real-world business processes. We refer the reader to the full specification of BPMN 2.0 [5] for the elements not covered in this chapter.

3.1 BPMN 2.0 – Business Process Modeling and Notation

BPMN, which stands for Business Process Model and Notation, is a diagramming language for specifying business processes in a business process model. It is based on a flowcharting technique very similar to activity diagrams in UML. This notation has been especially designed to coordinate the sequence of processes and messages that flow between participants in different activities [1].

Presently, many process modeling tools support BPMN for the reason that it is an industry standard, being maintained by the Object Management Group (OMG). That means it is not controlled by a single tool vendor. In [2], the developer of Bizagi BPM Suite describes why it is important to model with BPMN:

- BPMN is an internationally accepted process modeling standard.
- BPMN is independent of any process modeling methodology.
- BPMN creates a standardized bridge which reduces the gap between business processes and their implementation.
- BPMN enables you to model processes in a undefined and standardized way so that everyone in an organization can understand each other.

3.2 BPMN by Example

3.2.1 Order Process

Consider the process to deal with product ordering in Figure 3.1. The order process handles the necessary activities to receive, analyze and approve order requests submitted by customers. A

simplified version of this process consists of a few activities. First, the company recieves an order from customers, then the credit information submitted is verified. If approved, the process continues with the subprocess for order fulfillment, which checks whether a product is available or not. If available, an invoice will be sent and the order process will be completed.

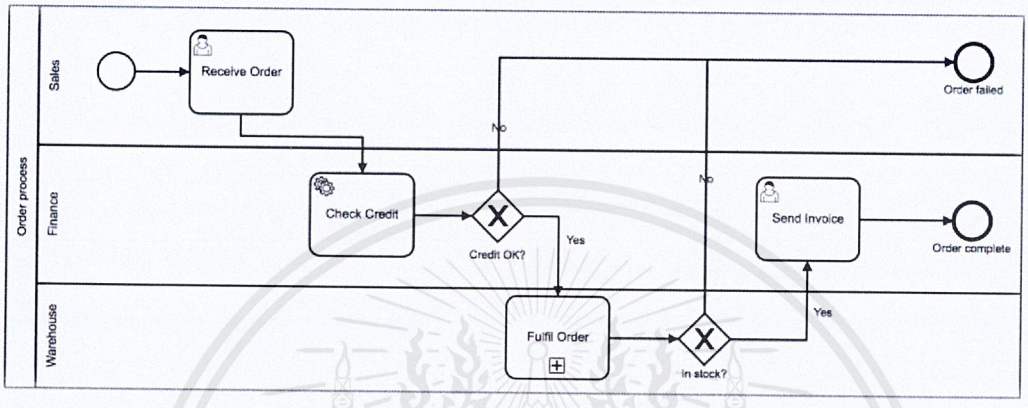

Figure 3.1. Order process

As you can see, within a diagram, there is a set of graphical elements that enables us to model a business process. The elements in the diagram are normally contained within an enclosing object called a pool with the name of the process written to the left of it. In case the process is carried out by more than one person or department, called actors, the pool can be divided into a number of swim lanes, each containing the tasks responsible by an actor. In our example, the pool contains the name of the process, which is "Order process", and three swim lanes for the Sales department, the Finance department and the Warehouse department.

From the diagram, we can identify three types of elements that describe the process behavior: the rounded rectangles, called activities, represent the works to be performed; the circles, called events, represent the starting point and the ending points of the process; and the diamonds, called gateways, represent the decision points in the process.

At the beginning, we indicate where and how a process starts by using a start event (thin circle). Processes can be started in many ways. BPMN defines a number of types of start events, for example, the simple start event, the message start event, the timer start event, etc., to model those behaviors. The rounded rectangles such as those labelled Receive Order, Check Credit, and Fulfill Order are activities. An activity represents an action, a specific unit of work to be performed. In this process, when the credit has been verified, different actions are to be performed depending on the result of the credit checking. This is represented by a branching point, called an exclusive gateway, in the diagram. In the diagram, if the result of credit checking is OK, the fulfill order activity will be performed, if not the process will be finished, as signified by the thick circle, called an end event.

In addition, BPMN allows us to indicate the type of each activity using an icon. In Figure 3.1, "Receive order" and "Send invoice" are tasks that are to be performed by human or involve human interaction. They are called *user tasks* in BPMN. On the other hand, "Check credit" is an automated task (i.e. to be executed without human intervention), called a *service task* in BPMN. In the case of "Fulfill order", a little [+] marker signifies that the activity is a subprocess, i.e. that activity is itself a process. The detail of a subprocess can be embedded inside the subprocess element itself, called *expanded subprocess*, as shown in Figure 3.2, or be given in a separate diagram, in which case the subprocess element is a *collapsed* one, as shown in Figure 3.1.

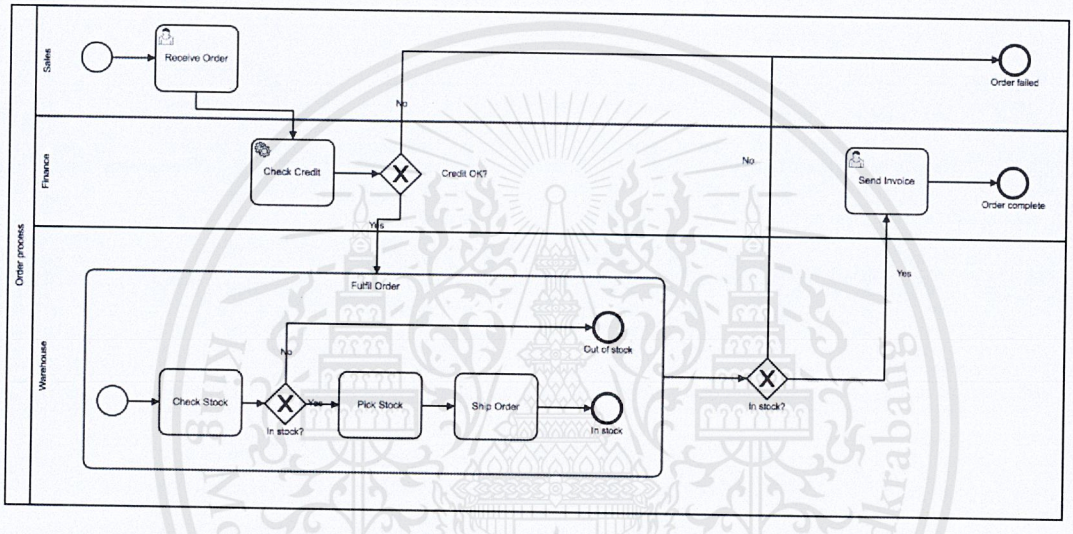

Figure 3.2. Order process including an expanded subprocess

From the description of the subprocess "Fulfill order" in Figure 3.2, the subprocess is triggered by the incoming sequence flow. This occurs when the result of "Check Credit" is OK. The sequence flow continues immediately from its start event. When the "Fulfill Order" subprocess completes, the process immediately continues following the sequence flow out of the subprocess. Note that the diagram for a subprocess may itself contain some other subprocesses. There is no limit to the number of levels you can nest subprocesses in this way.

3.2.2 Parallel Split and Join

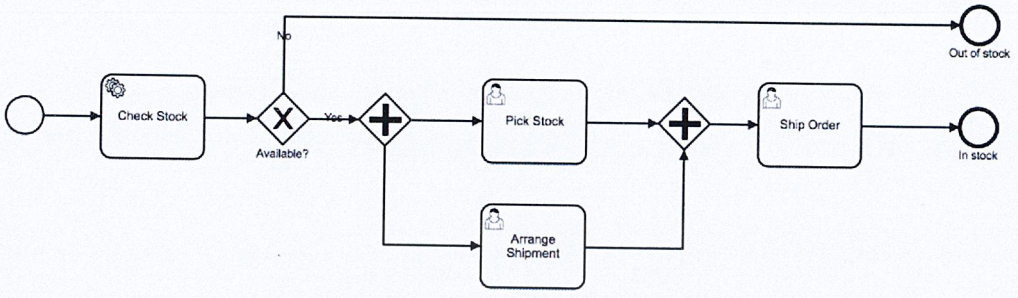

Figure 3.3. Parallel split and join

Figure 3.3 illustrates how both of parallel splits and joins look in a BPMN diagram. A gateway with a + inside is a parallel gateway, also called an AND-gateway. A parallel gateway with one sequence flow in and two or more flow out is called a parallel split or AND-split. As can be seen, both Pick Stock and Arrange Shipment are enabled to start at the same time as a result of the parallel split. The third gateway (left to right), with multiple sequence flows in and one out, is called an AND-join or synchronizing join. The process will wait at this gateway until all incoming sequence flows of the gateway have been executed.

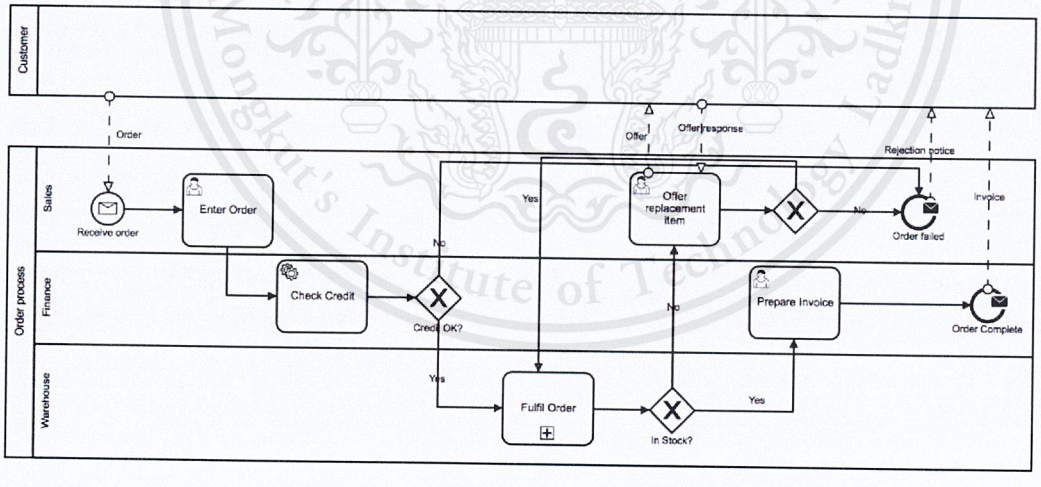

Figure 3.4. Order process collaboration diagram

3.2.3 Collaboration and Black-Box Pools

When considering shopping on an online store like Amazon.com, a customer might put some item in a shopping cart but, in the end, decide not to order it after all. Treating the customer as an external participant is the best way to solve this situation. In a BPMN diagram, it is recommended that an external participant be represented by a separate *black-box pool*. In the diagram in Figure 3.4, the

Customer pool is an example of a black-box pool. As shown, the Customer pool does not contain any element inside, since the Customer's internal process is invisible to us. The diagram in Figure 3.4 is called a *collaboration diagram* as it shows the interactions between the internal process and the external participant by means of message flows. The envelope symbol inside the start and end events reveal that these events receive and send message, respectively. A message start event signifies that a new instant of the process is created upon receipt of a message. On the other hand, a message end event signifies that the process sends a message when the end event is reached.

3.3 BPMN level 1 Palette

All of the symbols and shapes used in the previous section are part of the Level 1 palette, which is called the Descriptive Process Modeling Conformance subclass in BPMN 2.0[1]. The following is a complete list of the elements in the Level 1 palette.

3.3.1 Activity

An activity can be defined as a piece of work performed in a process. It is always represented by a rounded rectangle. The two basic activities are task and subprocess. A task is a type of activity that is considered to be atomic, meaning that it has no internal subparts described by the process model. On the other hand, a subprocess is an activity that can be decomposed hierarchically into other activities.

3.3.1.1 Task

A task is represented in the diagram by a rounded rectangle, with the task type indicated by a small icon in the upper left corner. The following is a complete list of the task types.

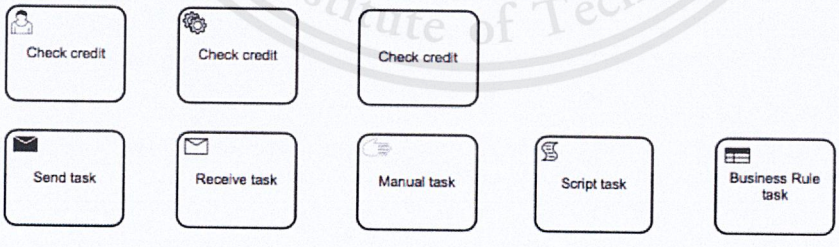

Figure 3.5. Top row, left to right: User task, Service task, Abstract task; Bottom row, left to right: Send task, Receive task, Manual task, Script task, Business Rule task

- A User task: a task performed by a human.
- A Service task: an automated activity.
- An Abstract task: the task type is undefined.

- A Send task: sends a message to an external participant on completion.
- A Receive task: waits for a message from an external participant.
- A Manual task: a task performed entirely manually.
- A Script task: a task executed by a business process engine.
- A Business Rule task: a task performed by a business rule engine.

3.3.1.2 Subprocess

A subprocess is a compounded activity, which means that it has subparts that can be described as a child-level process in the model. There are multiple ways of representing a subprocess in the diagram e.g. a subprocess with a [+] at the center of the bottom (Figure 3.6) is called as a collapsed subprocess. The internal process will start when the sequence flow reaches at the collapsed subprocess in the parent level. In contrast, when the execution of the internal process gets to the end event, it resumes on the sequence flow out of the collapsed subprocess in the parent level.

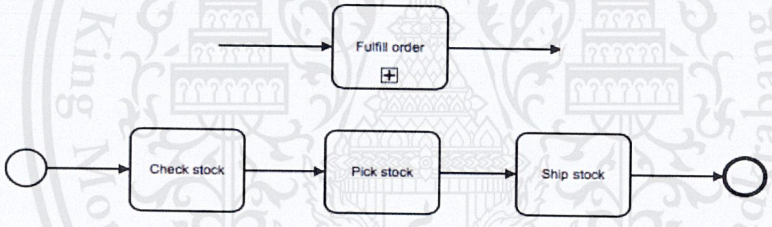

Figure 3.6. Hierarchical expansion: Collapsed subprocess in parent level (top) corresponds to child-level expansion (bottom) described in a separate diagram

3.3.2 Gateway

Gateways are special types of objects that instead of performing some activity simply control the flow through the process. BPMN defines several types of gateways, but the simplest and most commonly used are the *exclusive gateway* and the *parallel gateway*. An exclusive gateway, also called an *XOR gateway*, enables you to express decisions, while a parallel gateway, also known as an AND-gateway enables multiple threads of control through the process.

Figure 3.7. Left to right: Exclusive (XOR) gateway, Parallel (AND) gateway

3.3.3 Events

3.3.3.1 Start Event

A start event is represented as a circle with a single thin border. Its purpose is to indicate where and how a process or subprocess starts. BPMN 2.0 defines seven start event triggers, but the Level 1 palette includes only four of them, such as a Timer start event, which means that the process is triggered upon meet of a time condition, such as Weekly or Monday 9am.

Figure 3.8. Level 1 start events

3.3.3.2 End event

An end event is also represented as a circle similar to a start event but with the thick border. It is used to indicate the end of a path in a process or subprocess. BPMN 2.0 defines eight end event types, but the Level 1 palette includes only three of them, plus multiple end event. For example, a message end event signifies that a message is sent upon reaching the end event.

Figure 3.9. Level 1 end events

3.3.4 Connections

3.3.4.1 Sequence Flow

A solid line connector is called a *sequence flow*. It represents the sequential execution of process steps. When the node at the tail of a sequence flow completes execution, the node at the arrowhead is started. The only elements that can connect to the tail or head of a sequence flow are activities, gateways, and events. There are called flow nodes in BPMN 2.0.

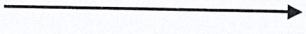

Figure 3.10. Sequence flow

3.3.4.2 Message Flow

A dash line connector in a diagram is called a *message flow*. It is used to represent the

communication between the process and an external entity. A message flow can be connected to any type of activity, a message event, or a black-box pool.

Figure 3.11. Message flow

3.3.5 Data Object and Data Store

Figure 3.12. Data object and data store

A data object represents data flow through a process. A data store represents persistance data, such as information store in the database. Both of them can be connected to other model elements through data associations as can be seen in Figure 3.13.

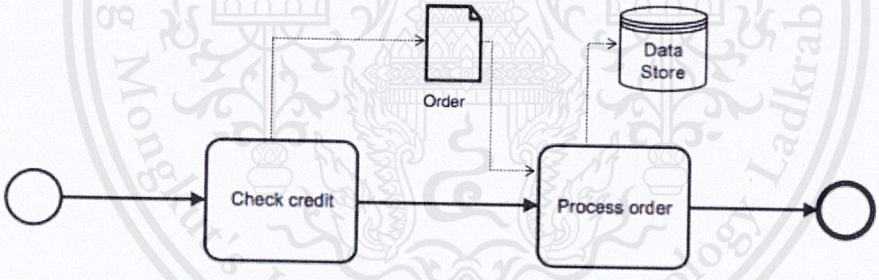

Figure 3.13. Data object and data store

3.4 BPMN Specification and Serialization

3.4.1 Process Modeling Conformance Class

According to the BPMN 2.0 specification, the Process Modeling Conference Class consists of three subclasses as follows:

- Descriptive subclass
- Analytic subclass
- Common executable subclass

3.4.1.1 Descriptive Subclass

The descriptive subclass corresponds to the level 1 palette. The elements and attributes in the table below are referenced by their XML names in the XSD.

Element	Attributes
participant (pool)	id, name, processRef
laneSet	id, lane with name, childLaneSet, flowElementRef
sequenceFlow	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
subProcess	id, name, flowElement
callActivity	id, name, calledElement
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

Figure 3.14. Descriptive subclass elements and attributes

The elements of the left column match up exactly with the level 1 palette. The right column is specified the details of each element that a tool must support in order to conform to the descriptive subclass.

3.4.1.2 Analytic Subclass

The analytic conformance subclass contains all of the elements of the descriptive conformance subclass plus the elements and attributes shown in Figure 3.15.

Element	Attributes
sequenceFlow	conditionExpression, default
sendTask	id, name
receiveTask	id, name
Looping Activity	standardLoopCharacteristic
Multi-instance activity	multiInstanceLoopCharacteristic
exclusiveGateway	Default
inclusiveGateway	id, name, default
eventBasedGateway	id, name, eventGatewayType
Link event pair	id, name, linkEventDefinition/@name
Signal start/end event	id, name, signalEventDefinition
Signal throw/catch intermediate event	id, name, signalEventDefinition
Signal boundary event cancelActivity	id, name, signalEventDefinition, attachedToRef,
Message throw/catch intermediate event	id, name, messageEventDefinition
Message boundary event cancelActivity	id, name, signalEventDefinition, attachedToRef,
Timer catching event	id, name, timerEventDefinition
Timer boundary event cancelActivity	id, name, timerEventDefinition, attachedToRef,
Error boundary event	id, name, errorEventDefinition, attachedToRef
Error end event	id, name, errorEventDefinition
Escalation throw intermediate event	id, name, escalationEventDefinition
Escalation end event	id, name, escalationEventDefinition
Escalation boundary event	id, name, escalationEventDefinition, attachedToRef, cancelActivity (false only)
Conditional start event	id, name, conditionalEventDefinition
Conditional catch intermediate	id, name, conditionalEventDefinition

event	
Conditional boundary event	id, name, conditionalEventDefinition, attachedToRef, cancelActivity
Message	id, name
Message flow	messageRef

Figure 3.15. Analytic subclass elements and attributes

3.4.1.3 Common Executable Subclass

In addition to the analytic and descriptive subclasses, the BPMN 2.0 specification enumerates the elements and attributes required to support executable BPMN, called the *common executable subclass*. In terms of the shapes and symbols included, common executable subclass is close to the descriptive subclass, but it includes additional child elements and attributes to specify the executable details. The common executable subclass requires support of XML schema as the type definition language, WSDL as the definition language for service interfaces, and XPath as the language for referencing data elements.

Element	**Attributes**
sequenceFlow	id, name, SourceRef, targetRef, conditionExpression, default
exclusiveGateway	id, name, gatewayDirection, default
parallelGateway	id, name, gatewayDirection
eventBasedGateway	id, name, gatewayDirection, eventGatewayType
userTask	id, name, rendering,implementation, resource, ioSpecification, dataInputAssociation, dataOutputAssociation, loopCharacteristic, boundaryEventRefs
serviceTask	id, name, implementation, OperationRef, ioSpecification, dataInputAssociation, dataOutputAssociation, loopCharacteristic, boundaryEventRefs
subProcess	id, name, flowElement, loopCharacteristic, boundaryEventRefs
callActivity	id, name, calledElement, ioSpecification, dataInputAssociation, dataOutputAssociation,

	loopCharacteristic, boundaryEventRefs
dataObject	id, name, isCollection, itemSubjectRef
textAnnotation	id, text
dataAssociation	id, name, SourceRef, targetRef, assignment
startEvent (None)	id, name
endEvent (None)	id, name
Message startEvent	id, name, messageEventDefinition (ref or contained), dataOutput, dataOutputAssociation
Message endEvent	id, name, messageEventDefinition (ref or contained), dataInput, dataInputAssociation
Terminate endEvent	id, name, terminateEventDefinition
Message	id, name, messageEventDefinition, dataOutput, dataOutputAssociation
Message intermediateThrowEvent	id, name, messageEventDefinition , dataInput, dataInputAssociation
Timer intermediateCatchEvent	id, name, timerEventDefinition
Error boundaryEvent	id, name, attachedToRef, errorEventDefinition, dataOutput, dataOutputAssociation

Figure 3.16. Common executable process modeling conformance subclass

The common executable subclass includes also the following supporting elements:

Element	**Attributes**
standardLoopCharacteristics	id, loopCondition
multiInstanceLoopCharacteristics	id, isSequential, loopDataInput, inputDataItem
rendering	
resource	id, name
resourceRole	id, resourceRef, resourceAssignmentExpression
ioSpecification	id, dataInput, dataOutput
dataInput	id, name, isCollection, itemSubjectRef
dataOutput	id, name, isCollection, itemSubjectRef
itemDefinition	id, structure (complexType) or import
operation	id, name, inMessageRef, outMessageRef, errorRef
message	id, name, structureRef
error	id, structureRef

assignment	id, form, to (complexType)
messageEventDefinition	id, messageRef, operationRef
terminateEventDefinition	id
timerEventDefinition	id, timeDate

Figure 3.17. Common executable subclass, supporting elements

3.4.2 BPMN Serialization

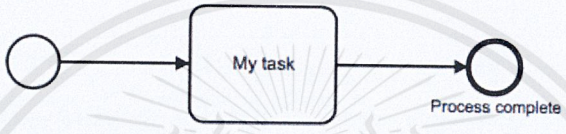

Figure 3.18. A simple process model

```
1  <definitions targetNamespace="https://www.itp-commerce.com">
2      xmlns="https://www.omg.org/spec/BPMN/20100524/MODEL"
3      xmlns:itp="https://www.itp-commerce.com/BPMN2.0"
4      xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
5      exporter="Process Modeler 5 for Microsoft Visio" exporterVersion="5.2742.13663 SR6"
6      itp:name="My Diagram" itp:version="1.0" itp:author="bruce" itp:createDate="8/2/2011 2:42:47 PM"
7      itp:modificationDate="8/3/2011 3:07:20 PM"
8      id="_4ab855a-76f3-4539-8a1d-60102f3b12e7">
9      <process id="_4188bfa1-cb2f-a84f-9f4f70b41a6b" name="My Process" processType="None">
10         <startEvent id="_3f808752-02dd-42d5-b4aa-2015031c7cc7"/>
11         <task id="_0532502d-31db-4fa5-920b-65c173652055" name="My task"/>
12         <endEvent id="_7986530a-fb47-4918-83fb-ad6c4f7d7656" name="Process completed"/>
13         <sequenceFlow id="6e913629-e553-47bf-875a-ce53cc167bdcc" sourceRef="_3f808752-02dd-42d5-
14         b4aa-2015031c7cc7" targetRef="_0532502d-31db-4fa5-920b-65c173652055"/>
15         <sequenceFlow id="_bfcd1cad-0c47-40df-9bd3-0e744bfe5bd2" sourceRef="_0532502d-31db-4fa5-920b-
16         65c173652055" targetRef="_7986530a-fb47-4918-83fb-ad6c4f7d7656"/>
17     </process>
18 </definitions>
```

Figure 3.19. Serialization of a simple process model

The standard serialization of a BPMN model is an XML document. Figure 3.15 shows a serialization of a simple model illustrated in Figure 3.14, generated by a proprietary tool that conforms to the BPMN 2.0 specification. The targetNamespace declaration from the first line is not model-specific, but is the same for all models serialized by this tool. The second line indicates that the default namespace is declared to be the BPMN 2.0 namespace. Additionally, two other namespaces as shown in Lines 3 and 4 are declared, the prefix xsi to reference the schemaLocation attribute and the prefix itp to reference tool vendor proprietary elements and attributes. The xsi:schemaLocation attribute indicates that this instance document is to be

validated against the BPMN 2.0 schema found at the relative file location schemas/BPMN20.xsd. The exporter and exporterVersion from Line 5 identify the tool and version used to create the serialization. Vendor proprietary attributes beginning with itp prefix are used to hold non-standard information about the model, such as the model name, version, author, creation date and modification date. Lines 9 – 17 contains the description of the elements in the diagram for this process. First, it must be noted that each element in the process is assigned an id that is automatically generated by the tool and is globally unique. Line 9 states that the process element has a processType value of None. Since this is the default value, this attribute could have been omitted altogether. The name attribute of each task and endEvent elements are matched with their labels in the diagram. Each sequenceFlow element's sourceRef and targetRef values are also matched with the id of the source and target nodes of the sequence flow, respectively.

Chapter 4
Requirements and Analysis

This chapter can be divided into two main sections and all of the following sections intend to give an explanation on software analysis. Section 4.1 discusses about requirements including both of functional requirements and non-functional requirements. Lastly, section 4.2 shows the use case diagram and also provides its descriptions.

4.1 Requirements

The requirements of our web-based workflow management system (WWFMS) can be summarized below:

4.1.1. Functional requirements

4.1.1.1 Overview

- WWFMS is a part of the Monkey Office project and will connect with other parts of the Monkey Office project during its operation.
- WWFMS is a web-based software application that can be used to describe common workflows within the International College, KMITL, automate that execution of such workflows.
- The description of a workflow is to be based on the BPMN 2.0 specification.

4.1.1.2 Users

- The users of the WWFMS are the users of the Monkey Office project.
- The user accounts are stored and managed by the user management component of the Monkey Office project, which is external to WWFMS.
- There is one special user account, called the admin, who has additional privileges on the system. The admin account of WWFMS is the same as the admin account of the Monkey Office project.

4.1.1.3 What the admin can do in WWFMS:

- The admin can create a description of a workflow by drawing diagrams in the BPMN 2.0 specification, and can modify/delete the created workflow descriptions.
- The admin can create/modify/delete web forms which are used to get information from the user or show information to the user when executing a workflow.
- The admin can create/modify/delete a customized task for use in a workflow description by providing a program in some scripting language for the task.
- For each workflow, the admin can specify the user role that is permitted to execute such a workflow.
- The admin can specify the user role responsible for each task within a workflow.
- The admin can see the list of all workflow executions, both finished and unfinished workflow executions, and their execution statuses.
- The admin can terminate the execution of a workflow initiated by any user.

4.1.1.4 What the user can do in WWFMS:

- The user can see the list of the available workflows that he/she can execute.
- The user can initiate an execution of a workflow.
- The user can see the list of all workflow executions initiated by him/her and their execution statuses.
- The user can terminate the execution of a workflow initiated by him/her.
- The user can interact with (e.g. providing input or obtaining information) a workflow execution through web forms.
- The user can see the list of the pending tasks in some workflow executions that require him/her to complete.
- The user can obtain a summary report of the workflow executions initiated by him/her.

4.1.1.5 What the system is required to do:

- The system shall support multiple workflow executions at the same time.

- In case a workflow execution requires an attention of a user, the system shall notify the user by email.
- The system supports file upload from the user when executing a workflow. The user shall be able to specify which user or user role can read/revise/delete each file uploaded.
- The system shall provide web services for other components of the Monkey Office project to
  - Retrieve the list of the workflows executable by the given user role
  - Retrieve the list of the finished workflow executions initiated by a user or by all users
  - Retrieve the list of the unfinished workflow executions initiated by a user or by all users
  - Retrieve the status of a workflow execution and the list of all the documents uploaded to or generated by that workflow execution
  - Retrieve the list of the pending tasks in unfinished workflow executions that require attention from the given user
  - Start the execution of a workflow (provided that the user currently logged on has a permission to do so)
  - Terminate the execution of a workflow (provided that the user currently logged on has a permission to do so)

4.1.2. Non-functional requirements

4.1.2.1 Implementation requirements:

- The system shall be a web-based software application with which the user may interact through a web browser.
- The system shall be implemented as a component of the Monkey Office project.
- The design of the user interface of the system shall be consistent with those of the other components in the Monkey Office project.
- The system shall, at least, support the following web browsers:

- On a Windows PC: Chrome (version 49), Firefox (version 45), and Internet Explorer (version 11)
- On a Mac computer: Chrome (version 49), Safari (version 9), and Firefox (version 45)
- On a mobile device: stock browser on iOS version 9.2 and stock browser on Android OS version 6.0

- The server-side component of the system shall run on a server computer running Ubuntu Linux version 15 with at least 2 Gigabytes of RAM available for use by the system.
- The continuous integration software development practice shall be adopted to minimize problems when integrating the system with the other components of the Monkey Office project.
- The design and implementation of the system shall be carried out in accordance with good software engineering practices. Documentations which facilitate debugging and further development shall be produced.

4.1.2.2 Capacity requirements

- The system shall be able to store and manage at least 1,000 workflows.
- The system shall support at least 10,000 concurrent executions of workflows.
- The system shall be able to keep the record of at least 10 million workflow executions.
- Each BPMN diagram describing a workflow may contain up to 1,000 components.
- Each web form may contain up to 1,000 static elements (i.e. elements added to the form during the design of the form).
- In each workflow execution, up to 1,000 files, totaling up to 10 Gigabytes, can be uploaded by the user.

4.1.2.3 Usability requirements

- The (non-admin) user who has no experiences in programming shall be capable to use the system after completing 6-hour training.
- The admin user who has basic programming skills shall be capable to use the

system after completing 20-hour training.

4.1.2.4 Efficiency and performance requirements

- Except for file upload, the system shall have a response time of no more than 3 seconds.
- The system is expected to run continuously for the indefinite period of time.

4.1.2.5 Security and privacy requirements

- The system shall not allow a user who has not logged on to the Monkey Office project to access the system.
- Only the admin has the permission to perform the tasks listed in Section 4.1.1.3 of this requirements.

4.2 Use case diagram

There are two actors within the use case diagram as shown below, namely, user and admin.

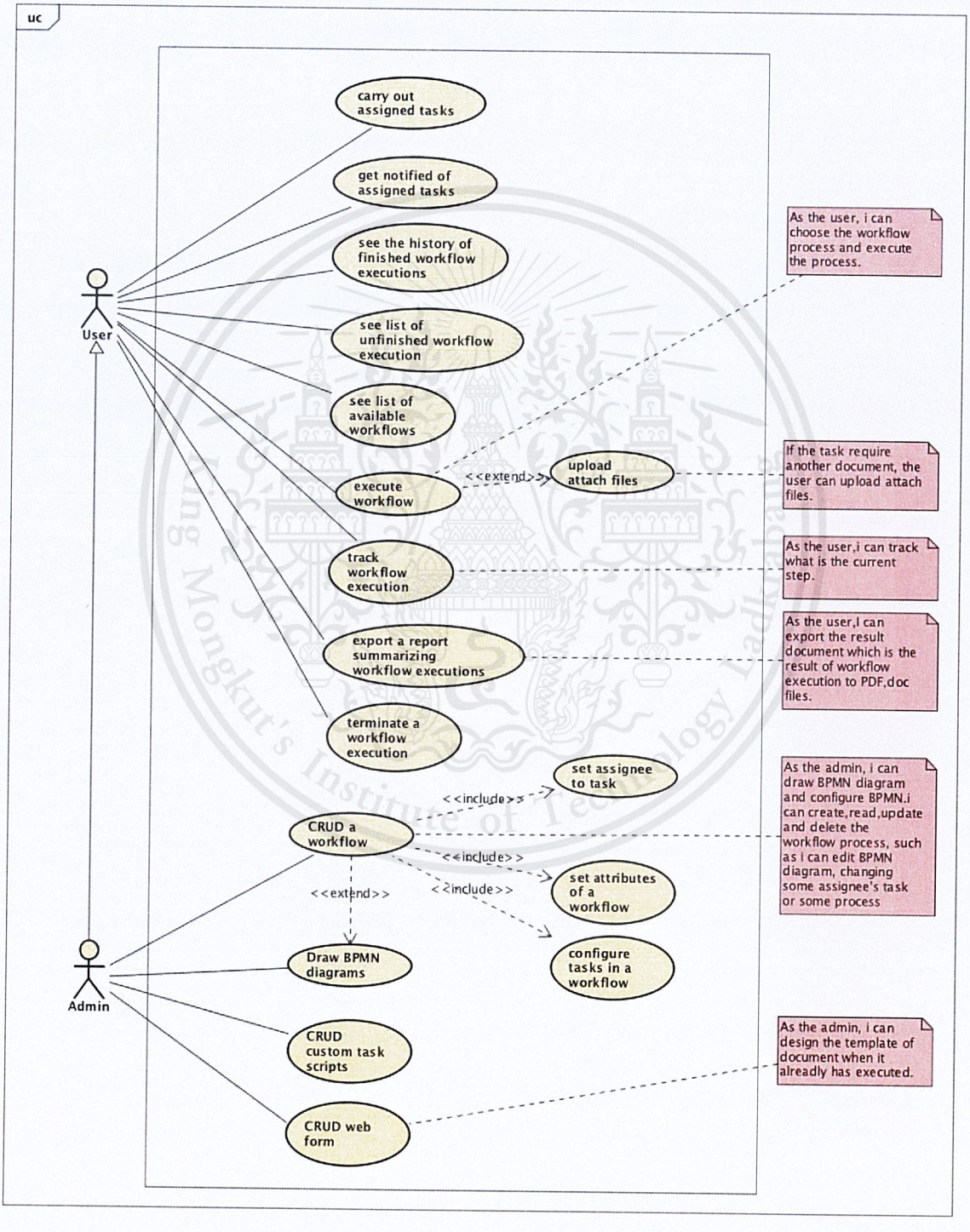

Figure 4.1. Use case diagram

4.2.1. Use case description

The following tables describe each use case in detail.

Use case:	Carry out assigned tasks	
Primary actor:	User	
Precondition	The user have been assigned to the task first.	
Overview	The user who are assigned to a task is able to carry out the assigned task.	
Flow of events	**Actor actions**	**System responses**
		1) the system will tell the user to do something like filling a information.
	2) User have to do following the system told like filling a information in the box.	
Alternative flow	-	-

Table 4.1. Use case description: carry out assigned tasks

Use case:	Get notified of assigned tasks	
Primary actor:	User	
Precondition	the tasks in that workflow have to execute first.	
Overview	The user who invoke with that workflow will get notified when they are assigned.	
Flow of events	**Actor actions**	**System responses**
	1) User choose workflow to execution.	
		2) The system will generate the website for the workflow.
		3) The system will send notification to the user when their tasks is running.
Alternative flow		3A) If the task which is running is not assigned that user, then the system will not send the notification to the user who is not assigned in the task.

Table 4.2. Use case description: get notified of assigned task

Use case:	See the history of finished workflow executions	
Primary actor:	User	
Precondition	-	
Overview	The user can see all list of their finished workflow executions.	
Flow of events	**Actor actions**	**System responses**
	1) User goes to finished history page.	
		2) The system will list all finished workflow executions of that user.
Alternative flow	-	2A) If the users do not have a list of finished workflow executions. Then it will be empty.

Table 4.3. Use case description: see the history of finished workflow executions

Use case:	See the history of unfinished workflow executions	
Primary actor:	User	
Precondition	-	
Overview	The user can see all list of their unfinished workflow executions.	
Flow of events	**Actor actions**	**System responses**
	1) User goes to unfinished history page.	
		2) The system will list all unfinished workflow executions of that user.
Alternative flow	-	2A) If the users do not have a list of unfinished workflow executions. Then it will be empty.

Table 4.4. Use case description: see the history of unfinished workflow executions

Use case:	See list of all available workflows	
Primary actor:	User	
Precondition	There must be workflows which created from the admin first.	
Overview	User can see list of available workflows which are available to execute.	
Flow of events	**Actor actions**	**System responses**
	1) User goes to executed workflow page.	
		2) The system will list all available workflows.
Alternative flow	-	2A) If the user do not have a list of workflows. Then it will be empty.

Table 4.5. Use case description: see list of all available workflow

Use case:	Execute workflow	
Primary actor:	User	
Precondition	There must be workflows which created from the admin first.	
Overview	User can select a workflow to execute.	
Flow of events	**Actor actions**	**System responses**
	1) User select a workflow.	
		2) The system will generate the website for the workflow.
	3) User have to respond to the website to continue process.	
Alternative flow	-	-

Table 4.6. Use case description: execute workflow

Use case:	Upload attach files	
Primary actor:	User	
Precondition	The user must to execute the workflow and the task in that workflow execution have to require document.	
Overview	The user can upload outside files or document.	
Flow of events	**Actor actions**	**System responses**
		1) When the system generates a website for the workflow and the task of workflow execution require some outside document system. Then the system will provide a upload attach files button.
	2) User have to click upload button	
	3) User have to choose files from their computer.	
		4) The system will keep those document in the system.
Alternative flow	-	1A) If there are no task require for upload the outside document. Then there are no a upload attach files button.

Table 4.7. Use case description: upload attach files

Use case:	Export a report summarizing workflow executions.	
Primary actor:	User	
Precondition	To export a report, that workflow executions have to be done first.	
Overview	User can export a report.	
Flow of events	**Actor actions**	**System responses**
	1) User goes to finished history page.	
	2) User select the finished workflow execution which one that they want to export to a report.	
		3) The system will generate the report summarizing workflow executions.
Alternative flow	-	-

Table 4.8. Use case description: export a report summarizing workflow executions

Use case:	Track workflow execution	
Primary actor:	User	
Precondition	There must be workflows which is executing by user first.	
Overview	User can track their workflow execution.	
Flow of events	**Actor actions**	**System responses**
	1) User goes to "doing page"	
		2) The system will list their workflow execution and it's current task.
	3) If there is the task which is assigned to the user to do it. So, the user can click to continue the workflow execution.	
Alternative flow	3A) If there is not the task which is assigned to the user to do it. So, the user can not do anything until their task coming.	-

Table 4.9. Use case description: track workflow execution

Use case:	Terminate a workflow executions	
Primary actor:	User	
Precondition	User have to be executing a workflow.	
Overview	User can cancel the workflow executions	
Flow of events	**Actor actions**	**System responses**
	1) User have to click cancel button in a workflow execution.	
		2) The system will terminate the workflow execution immediately.
Alternative flow	-	-

Table 4.10. Use case description: terminate a workflow executions

Use case:	CRUD a workflow	
Primary actor:	Admin	
Precondition	-	
Overview	Admin can create, read, update and delete workflows.	
Flow of events	**Actor actions**	**System responses**
	1) Admin have to go to list of available workflows.	
	2) Admin can click create workflow to create a new one.	
		3) The system will link to create workflow page.
Alternative flow	2A) Admin can click the old workflow to edit it.	
		3A) The system will open that workflow page and allow the admin to edit it.

Table 4.11. Use case description: CRUD a workflow

Use case:	Set assignee to task	
Primary actor:	Admin	
Precondition	During admin is creating or editing a workflow	
Overview	Admin can assign a person role who take responsible to that task.	
Flow of events	**Actor actions**	**System responses**
	1) Admin have to click to that task and assign the person role to the task.	
		2) The system will save it in database.
Alternative flow	-	-

Table 4.12. Use case description: set assignee task

Use case:	Set attributes of a workflow.	
Primary actor:	Admin	
Precondition	During admin is creating or editing a workflow.	
Overview	Admin can define all attributes of a workflow.	
Flow of events	**Actor actions**	**System responses**
	1) Admin can add attributes of a workflow.	
		2) The system will save it in database.
Alternative flow	-	-

Table 4.13. Use case description: set attributes of a workflow

Use case:	Configure tasks in a workflow	
Primary actor:	Admin	
Precondition	During admin is creating or editing a workflow.	
Overview	Admin can configure all tasks in a workflow.	
Flow of events	**Actor actions**	**System responses**
	1) Admin configure each tasks in a workflow and configure the workflow.	
		2) The system will record it in database.
Alternative flow	-	-

Table 4.14. Use case description: configue task in a workflow

Use case:	Draw BPMN	
Primary actor:	Admin	
Precondition	During admin is creating or editing a workflow.	
Overview	Admin can draw a Business Process Model and Notation for a workflow.	
Flow of events	**Actor actions**	**System responses**
	1) Admin can drag and drop the attribute of BPMN to draw a BPMN diagram.	
		2) The system will save it in database.
Alternative flow	-	-

Table 4.15. Use case description: draw BPMN

Use case:	CRUD custom task script.	
Primary actor:	Admin	
Precondition	During admin is creating or editing a workflow.	
Overview	Admin can create, read, update and delete a custom task script.	
Flow of events	**Actor actions**	**System responses**
	1) Admin can select a script task and create a new custom task service for a script task.	
		2) The system will save it in database.
Alternative flow	1A) If there is a script service that admin would like to use. Admin can use it.	-

Table 4.16. Use case description: CRUD customer task script

Use case:	CRUD web form	
Primary actor:	Admin	
Precondition	During admin is creating or editing a workflow.	
Overview	Admin can create, read, update and delete a web form.	
Flow of events	**Actor actions**	**System responses**
	1) Admin can select user task and create a new form for a user task.	
		2) The system will save it in database.
Alternative flow	1A) If there is form that admin would like to use for that user task. Then admin can use that form.	-

Table 4.17. Use case description: CRUD web form

Chapter 5
Software Design

This chapter can be divided into four major parts. All parts are intended to provide an explanation on software design in detail. Initially, Section 5.1 shows the overall architecture of the system. Section 5.2 describes the user role management. Data management is explained in section 5.3. Finally, Section 5.4 explains workflow exeuction.

5.1 Overall architecture

The architecture of the entire system can be described in Figure 5.1.

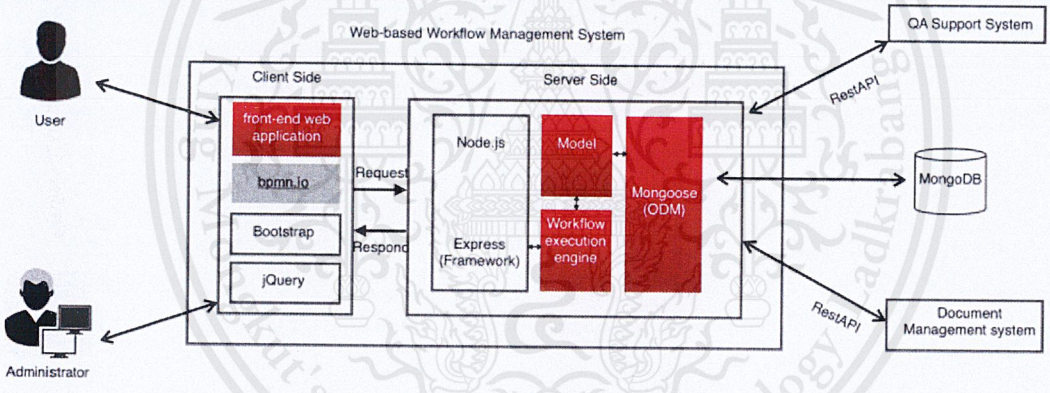

Figure 5.1. System architecture

The whole system consists of the client side and the server side.

Client side: The part of our software that runs on the user's computer and connects to the server. In most cases, the client side runs on a web browser. The client side consists of four parts as follows:

- **Front end web application** is a part that include many modules of javascript. The overall benefits are aimed to use for making our system easily to use and power functionality.
- **jQuery** is a library that provides an easier method of writing a script on client side.
- **Bootstrap** is group of HTML and CSS that intend to provide a comfortable way for

designing a user interface.

- **bpmn.io** is an open-source software library that simplifies creating, embedding and extending BPMN diagrams.

Server side: There are two major parts of the server side, namely the web application and the database management system.

- **Web application:** Node.js is used to implement and Express is used to make a web framework.
- **Database:** *MongoDB* is used as our database management system. In our development, the MongoDB server that we used was hosted on a cloud service provider called *mLab*. In addition to this, we employ a javascript software module called *mongoose* which acts as an interface between our web application and MondoDB server to support object-oriented data modelling for our project.
- **Workflow execution engine** is used to process the elements in the BPMN diagram.
- **Model** is a collection of schemas for data objects.

5.2 User role management

There are two main kinds of user roles within the MonkeyOffice system, namely simple role and relative role. A simple role is a group of users. For example, the role Student consists of all the users who are students. A relative role is a binary relation between users. For example, the role Supervisor defines which user is a supervisor of which user.

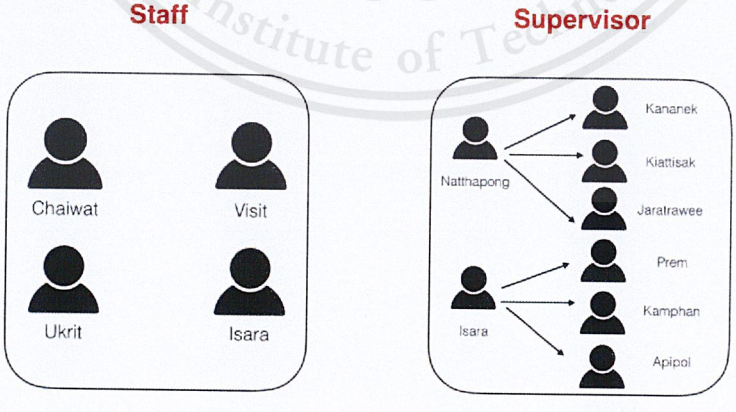

Figure 5.2. User role management

5.3 Data management

5.3.1 Data model

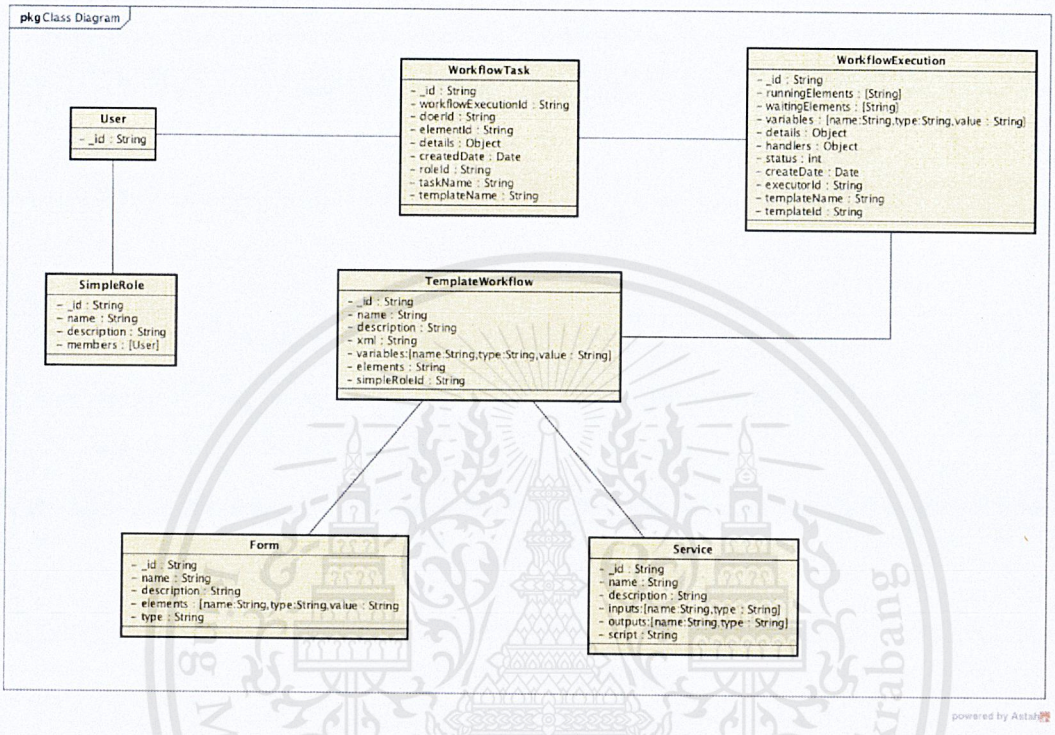

Figure 5.2. Database diagram

5.3.2 Temporary data

Temporary data is the data created and maintained by the system while a workflow is executing. Once the execution of the workflow is finished, the temporary data will be deleted. In our workflow management system, the temporary data includes the state of the execution of the workflow and the information stored in variables. The use of variables was introduced in a workflow management system called Bonita (developed by Bonitasoft Inc.). Variables are used to store the information generated during the execution of a workflow. For example, the information provided by the user through a form may be first temporarily stored in variables before eventually saved into the database. Variables also allow the information created or obtained from one task to be transferred to or shared with another task. In our system, there are two main kinds of variables:

Workflow variable are variables that store information for the entire period of workflow execution;

Task variable are variables that store the information during the execution of a task. There are two types of task variables:

- For user tasks, **form variables** are used to pass the information from the workflow execution to the form and the information received on the form back to the workflow execution.
- For service tasks, **script variables** are used to pass the information from the workflow execution to the script and the information generated from the script back to the workflow execution.

Data mapping. To enable the data generated or obtained from each task to be used elsewhere in the workflow, a mapping between task variables and workflow variables can be specified for each task. An input mapping for a task allows the information to be sent from the workflow variables to the task variables before the task starts execution. On the other hand, an output mapping allows the information to be sent from the task variables back to the workflow variables after the task finishes execution.

5.3.3 Persistent data

Persistent data is the data that is maintained even after the workflow execution has terminated. In our system, persistent data is stored in a database. To make the use of persistent data more convenient, instead of having the workflow designer writing code to store persistent data in the database by him/herself, we adopt the idea of workflow data model, which is based on a similarly-named concept in the workflow management software Bonita.

A workflow data model is basically a collection of schemas for data objects. In addition to the fields in the data objects, each schema also includes methods for retrieving data from the database, as well as methods for updating the records in the database. For instance, the administrator may define the method `Student.findAll(sid)` by a database query "`SELECT * FROM STUDENTS WHERE id = {sid}`". The workflow designer can then retrieve student records using the method `Student.findAll` instead of having to write a script for database query by him/herself.

5.4 Workflow execution

5.4.1 Workflow execution engine diagram

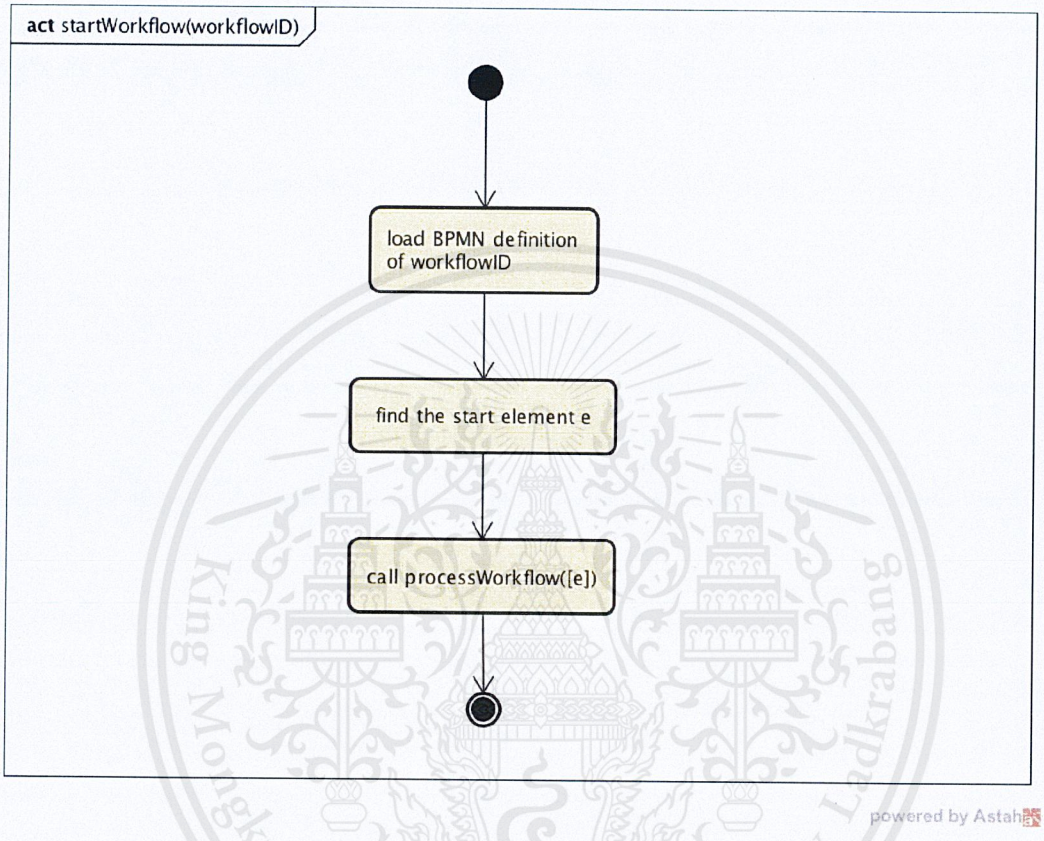

Figure 5.3. Flowchart diagram of workflow execution engine

When the user executes the workflow, the system will load the BPMN diagram from the workflowID and find the start element e. Then call processWorkflow([e]).

Figure 5.4. Execution of each element

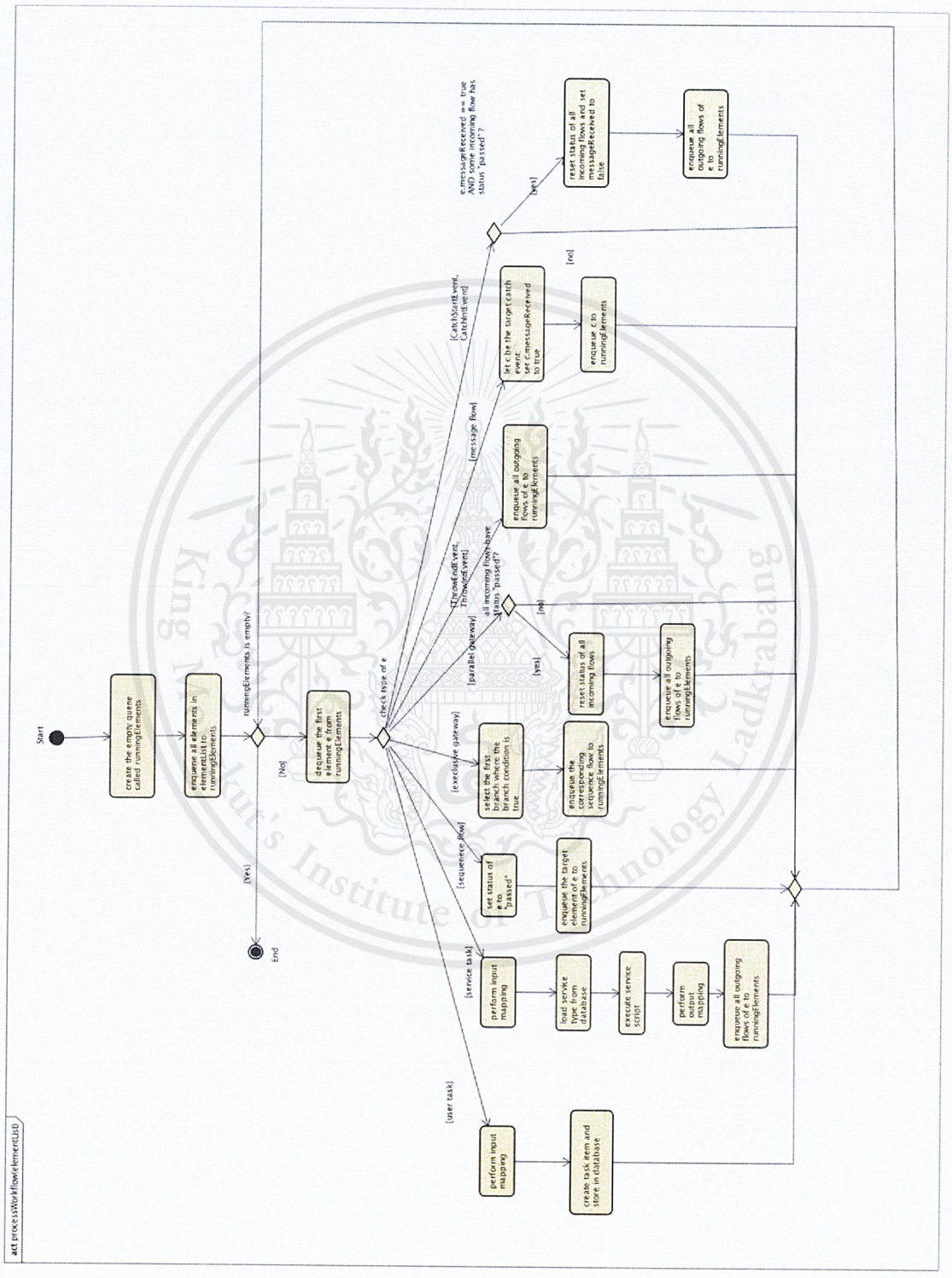

Figure 5.4 describes the execution of each element. The processWorkflow maintain a queue named runningElements, which contains the elements in the BPMN diagram. Initially, the engine will check whether the queue is empty or not. If not, the first element in the queue is taken out. Execution engine will check the elements type and choose an appropriate action for that element depending on its type. Then the execution engine goes back to retrieve the next element on the the queue runningElements. If the queue is empty, the engine will update the execution status including the values of each workflow variable.

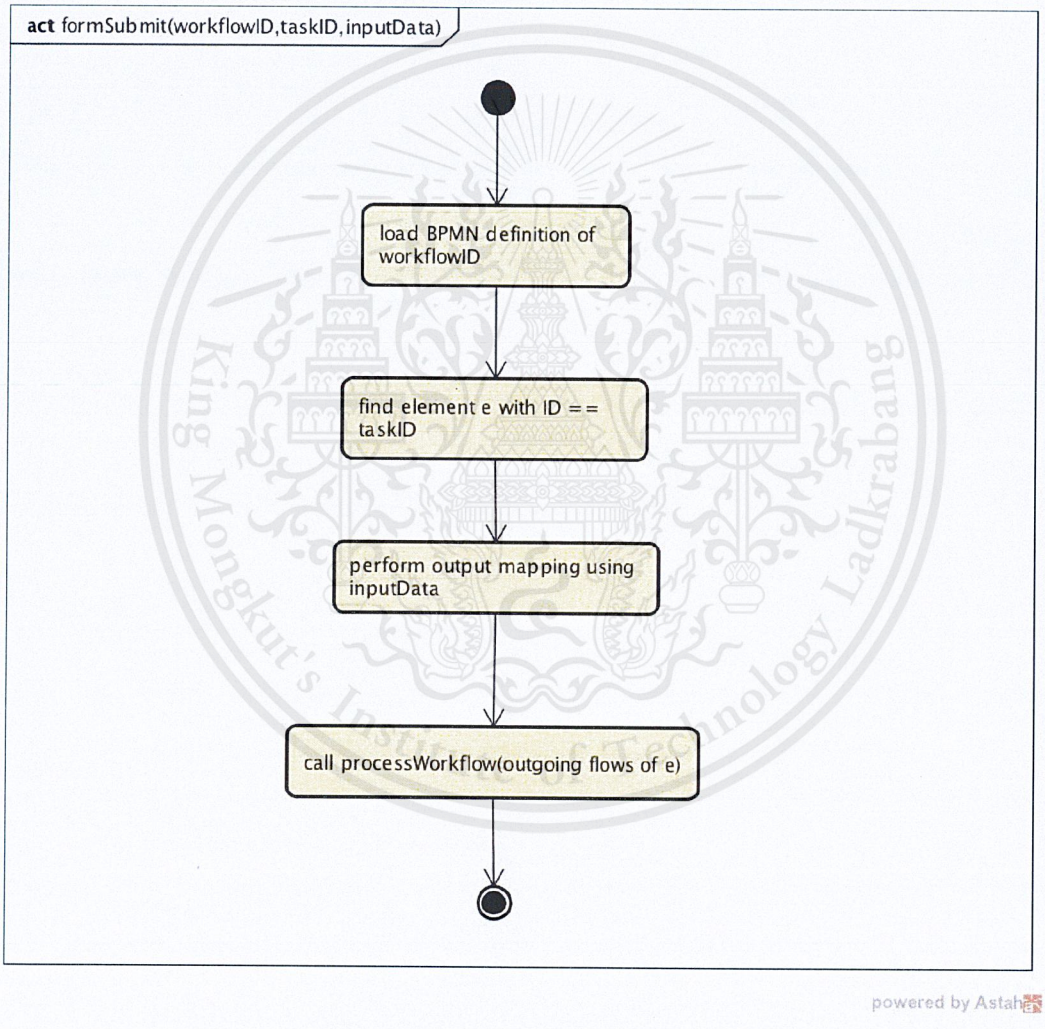

Figure 5.5. Form submit diagram

As can be seen in Figure 5.5, once the user clicks a submit button on a the form that associated with a user task, the engine will load BPMN definition of workflowID and find element e whose ID equals to taskID. The next step is to perform the output mapping by using the data returned form the form. Finally, call processWorkflow(outgoing flow of e).

Chapter 6
Development

This project is developed using Node.js and Express.js as the primary development framework and MongoDB as the database management system. This chapter can be divided into four major sections. First of all Section 6.1 provides a description on Node.js and MongDB, and then Section 6.2 explains some development techniques used in this project. Section 6.3 gives a comparison of MongoDB and relational databases. Lastly, Section 6.4 provides some essential development techniques.

6.1 Node.js

Node.js is an open-source, cross-platform runtime environment for developing server side web applications. It has many supporting modules written in JavaScript. Node.js is built upon Chrome's V8 JavaScript engine. It adopts an event-driven architecture and a non-blocking I/O model. Node.js has a reputation for its speed and scalability. The primary reasons for its performance are the following:

- Asynchronous event-driven architecture
- Single-thread, with non-blocking I/O
- Efficient implementation

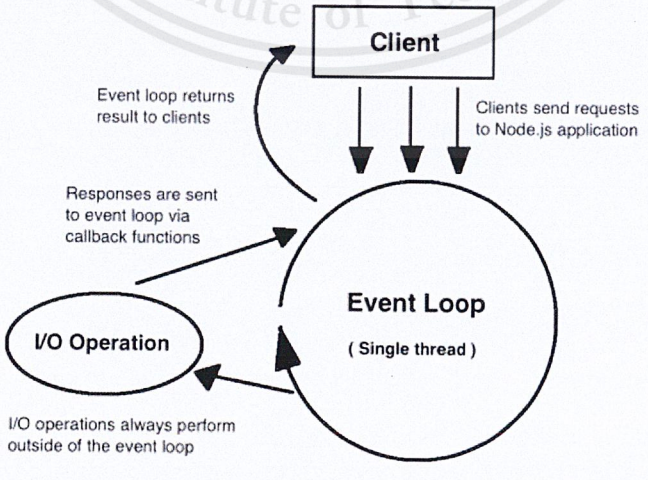

Figure 6.1. How Node.js works

A Node.js application uses a single thread which runs the main event loop to handle all client requests. When an I/O operation needs to be performed, it is executed outside of the main event loop. When the I/O operation is finished, it will send a response to the main event loop via a callback function. Because I/O operations are always performed outside of the main event loop, this allows the main event loop to respond to other client requests without waiting for the I/O operations to complete.

6.2 MongoDB

MongoDB is a free, open-source, document-based database management system designed for ease of development and scaling. It is classified as a NoSQL database system. MongoDB collects the data in the form of JSON-like documents with dynamic schemas.

Figure 6.2. Example of a JSON-like document

MongoDB supports the modeling of relationships between data elements in a database, but not in the same way as a relational database system does. There are two methods to model relationships in MongoDB: embedded relationships and referenced relationships.

6.2.1 Modeling Embedded Relationships

In this method, referenced data is embedded into the referencing document.

Figure 6.3. Example of embedded documents in MongoDB

6.2.2 Modeling Referenced Relationships

In this method, referenced data is stored in separate documents. The referencing document contains links to the referenced documents.

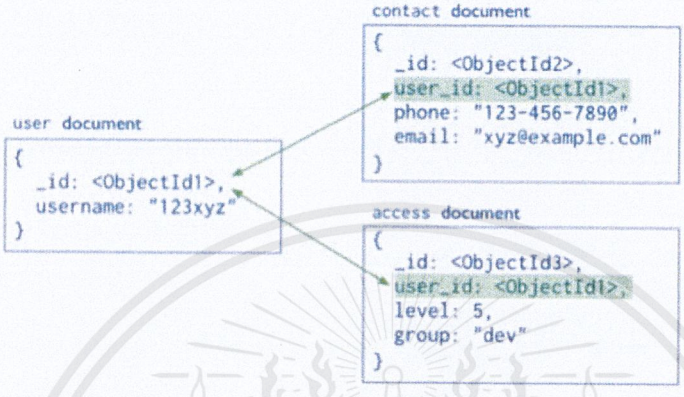

Figure 6.4. Example of referenced relationships in MongoDB

6.3 Comparison of MongoDB and Relational Databases

In recent years, there are 2 types of database management systems that are widely used: relational database systems, such as Oracle or MySQL, and non-relational database systems, such as MongoDB. Relational databases represent data in tables. Document-based databases, such as those stored in MongoDB, represent in JSON-like documents. In relational databases, we need to define table schemas, i.e. the list of columns. Every row in a table must have the same columns. On the contrary, in MongoDB, we do not need specify the fields of each data item beforehand, also data items in a collection can have different fields.

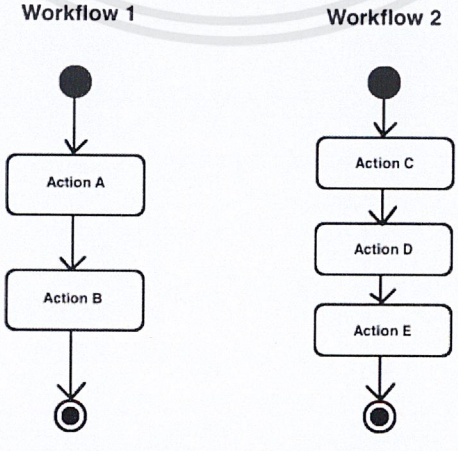

Figure 6.5. Example of workflows

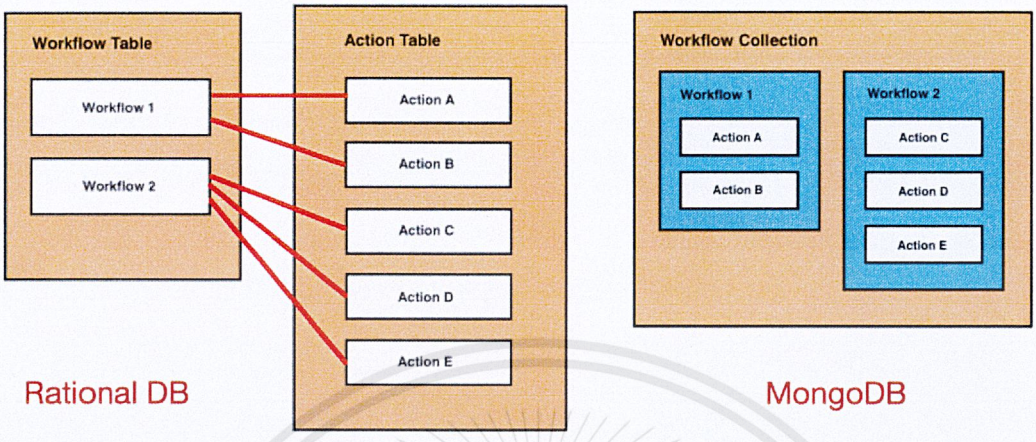

Figure 6.6. Example how to store data between relational database and mongoDB

6.4 Development Techniques

6.4.1 Promise

Promise is a built-in feature in JavaScript. Promise is an object that represents the value that may not be available yet, but will be used at some point in the future. We use this feature in the part of the workflow execution engine in our project. This is shown in Figure 6.7.

```
function mainLoop(runningElements, execution, callback){
    if( runningElements.length > 0 ){
        var elementId = runningElements.splice(0,1)[0];
        doElement(elementId, execution)
        .then(postAction)
        .then(function(){
            mainLoop(runningElements, execution, callback);
        })
        .catch(function(error){
            callback(error);
        });
    }
    else{
        callback(null);
    }
}
```

Figure 6.7. The main loop of the wirktlow execution engine

In figure 6.7, the code in `doElement()` and `postAction()` functions run asynchronously with workflow execution engine. To make them run sequentially (i.e. ensuring that `postAction()` starts only after `doElement()` finishes), we use promise in JavaScript.

Chapter 7
Result

We are responsible for both the client side and the server side of the web application. We have implemented our web application with an easy-to-use user interface as well as providing an essential set of features, including creating and modify BPMN diagrams. In this section, we focus on describing the user interface in detail. To see our Web-based Workflow Management System Web Application in action, please visit the following URL: https://monkeyoffice.herokuapp.com

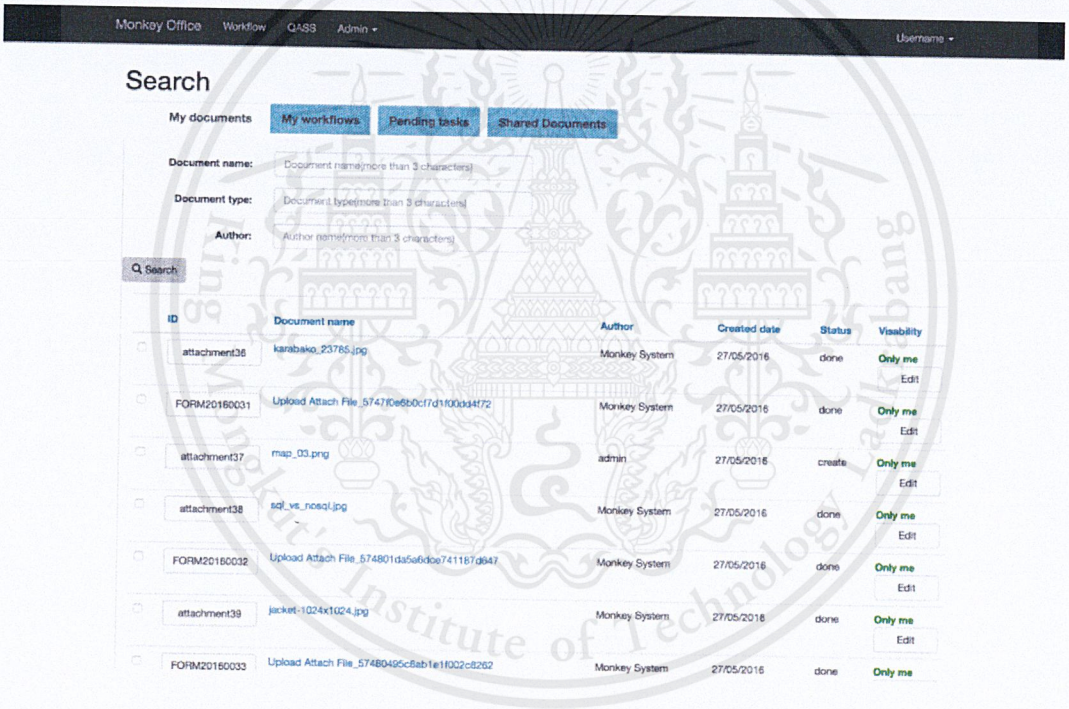

Figure 7.1. Main page of the web application

Figure 7.1 shows the main page of our web application. There is a workflow navigator in the top of our web application. In the Admin dropdown on the workflow navigator, there is a menu item for creating a worklfow. In the middle of the page, there is a table which shows the documents, the workflow execution history, and the list of pending tasks of that user.

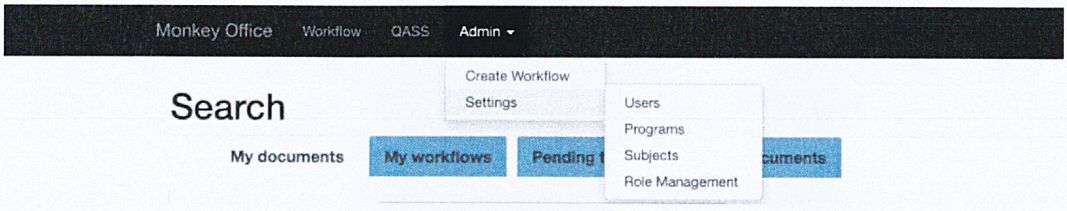

Figure 7.2. Navigation bar

There are two main tap that we are focus in Figure 7.2 which are related about workflow, first feature is create workflow and second feature is workflow which show the list of workflow template list.

7.1 Workflow creation

When the user clicks on 'Create Workflow', the web site will direct us to the workflow creation page. The user can draw a workflow on that page as shown in Figure 7.3.

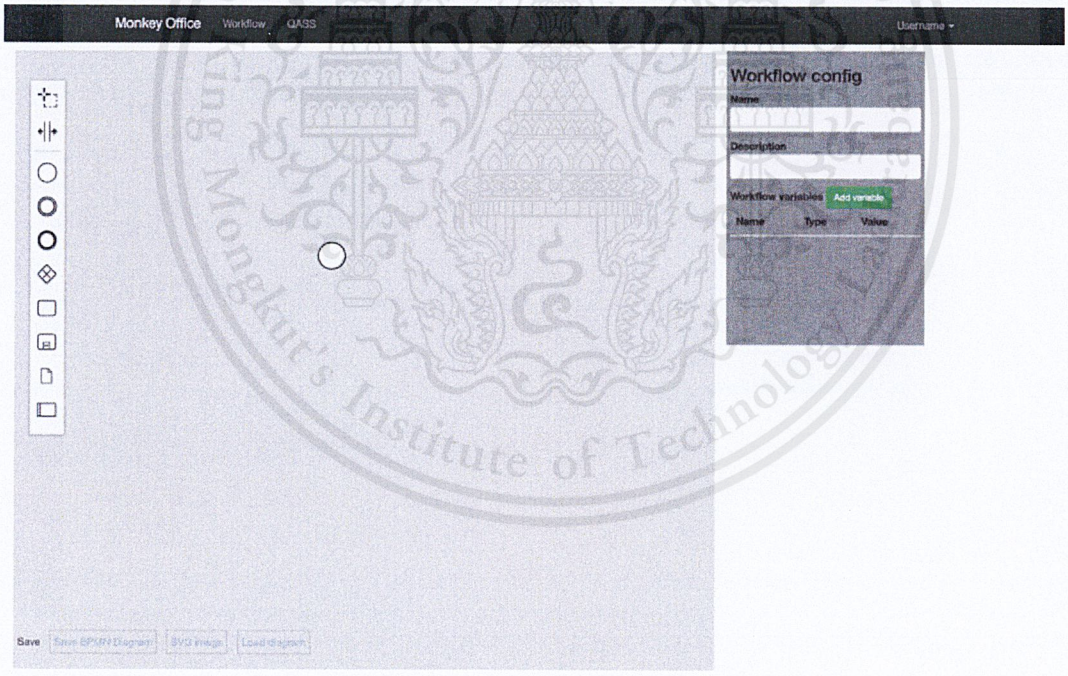

Figure 7.3. Workflow creation

As seen in Figure 7.3, the user can draw a diagram by dragging the elements of BPMN from the toolbox on the left and drop the elements in the center area. The user can set or adjust the attribute of each BPMN element in the diagram from the workflow configuration panel on the left of the page.

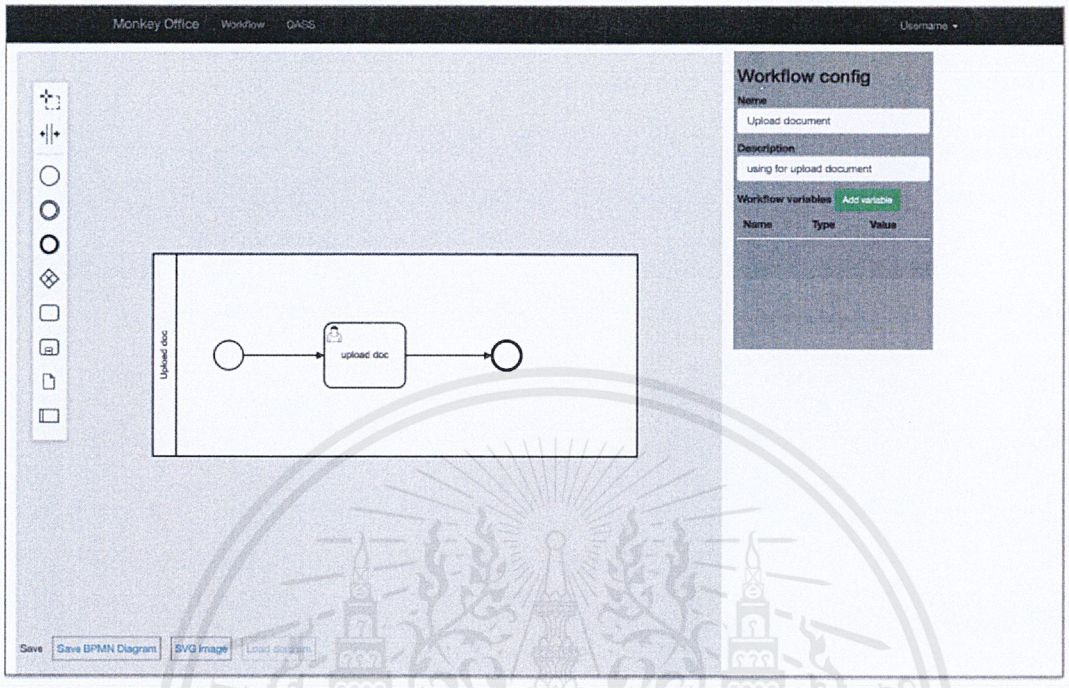

Figure 7.4. Create a workflow for uploading a document

In Figure 7.4, the configuraiton panel on the right allows the creator to set the name and the description of the workflow. It also allows the creator to define workflow variables for the workflow.

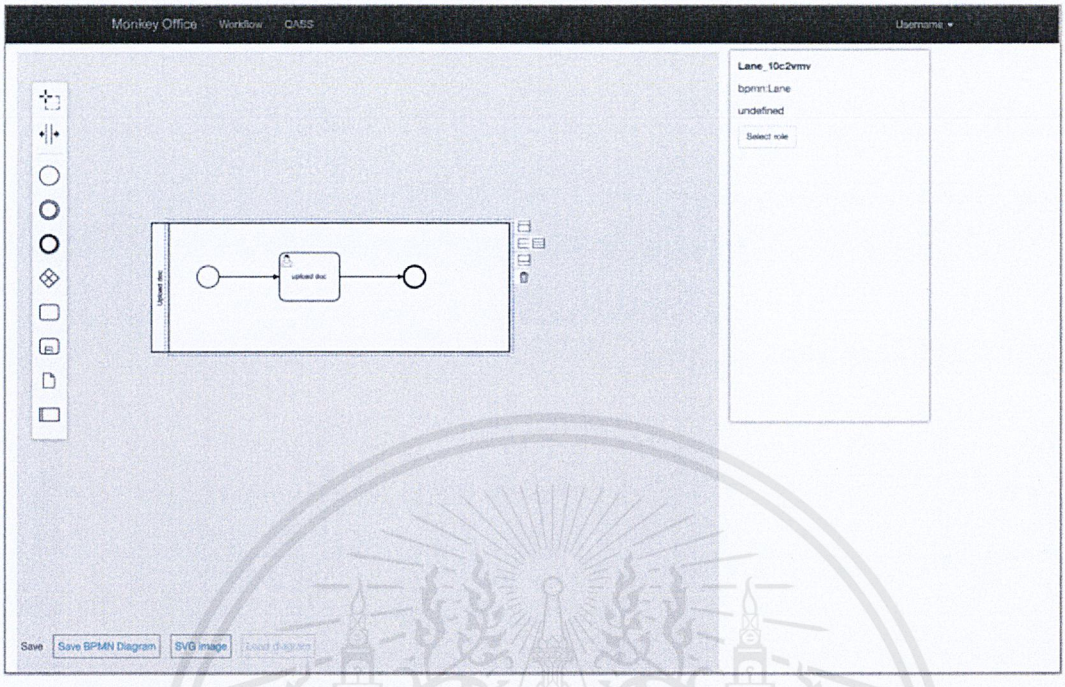

Figure 7.5. Configure the role of the users who can perform this workflow

As shown in Figure 7.5, we can specify the role of the users in which this workflow is for. Such users will be given the permission to execute this worklfow.

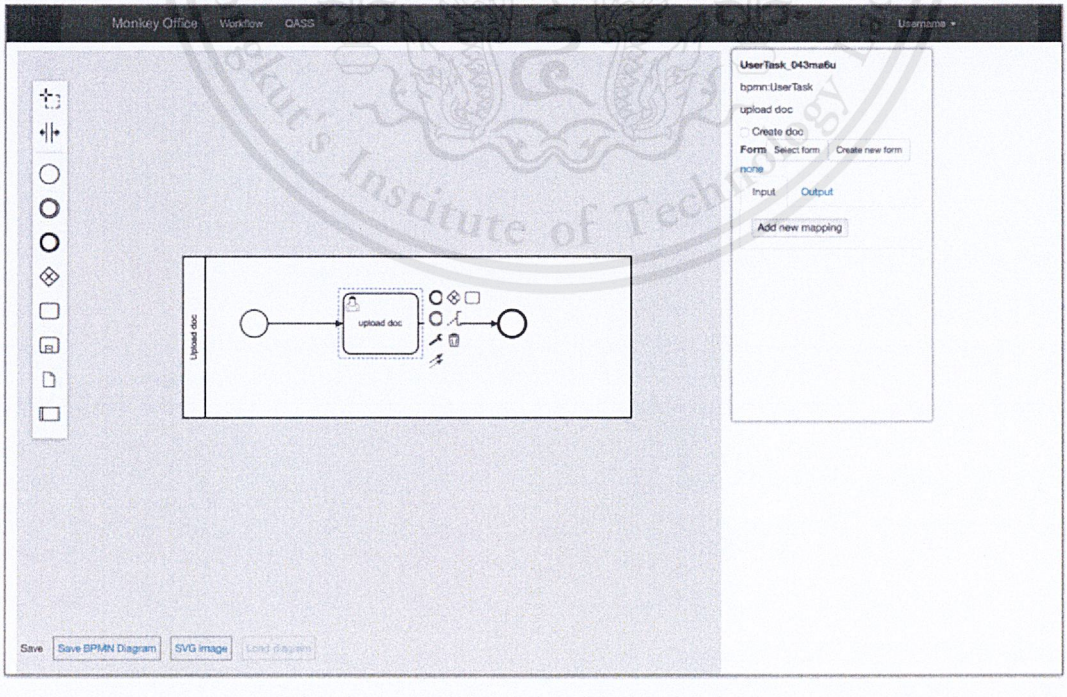

Figure 7.6. Task configuration in the workflow creation page

Figure 7.6 shows the configuration panel for a user task. The workflow creator can select a form to associate with the user task.

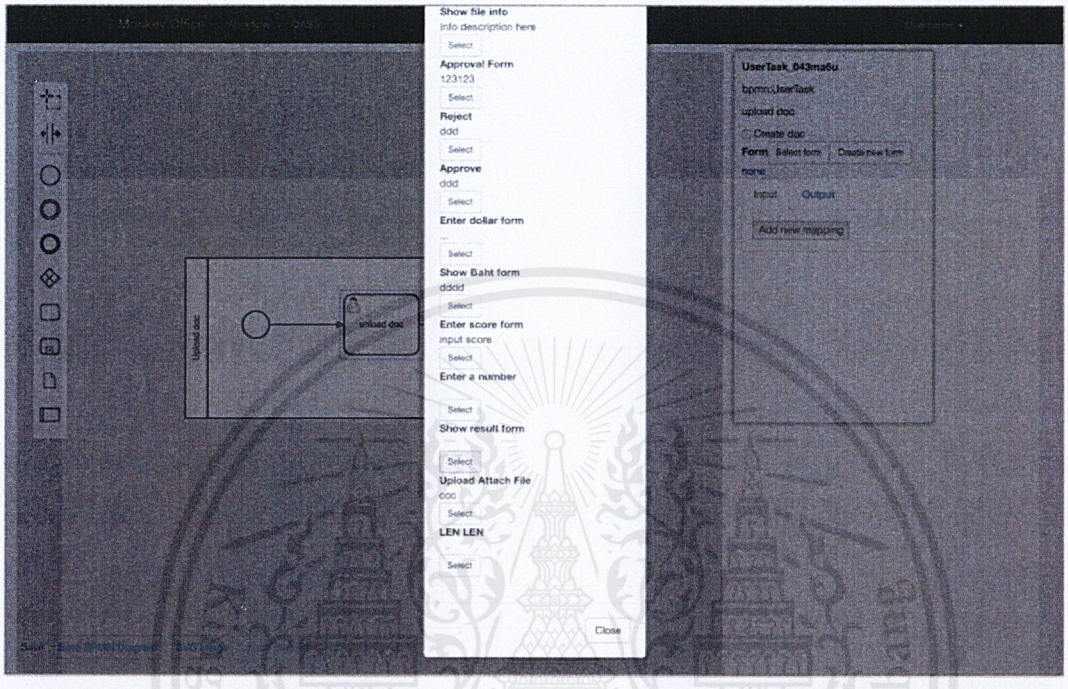

Figure 7.7. Form selection page

In Figure 7.7, we select the form for upload an attachment file from the list. Since this form does not require input data and does not return any output data, the input and output mapping are left blank.

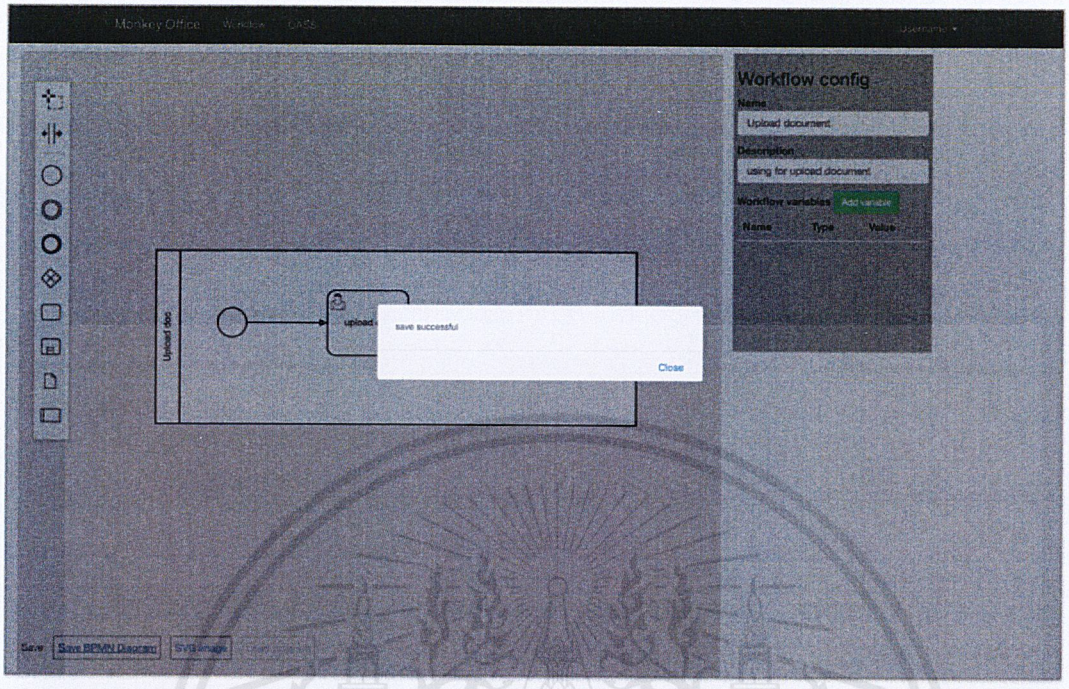

Figure 7.8. Saving the created workflow

Figure 7.9. List of workflow templates

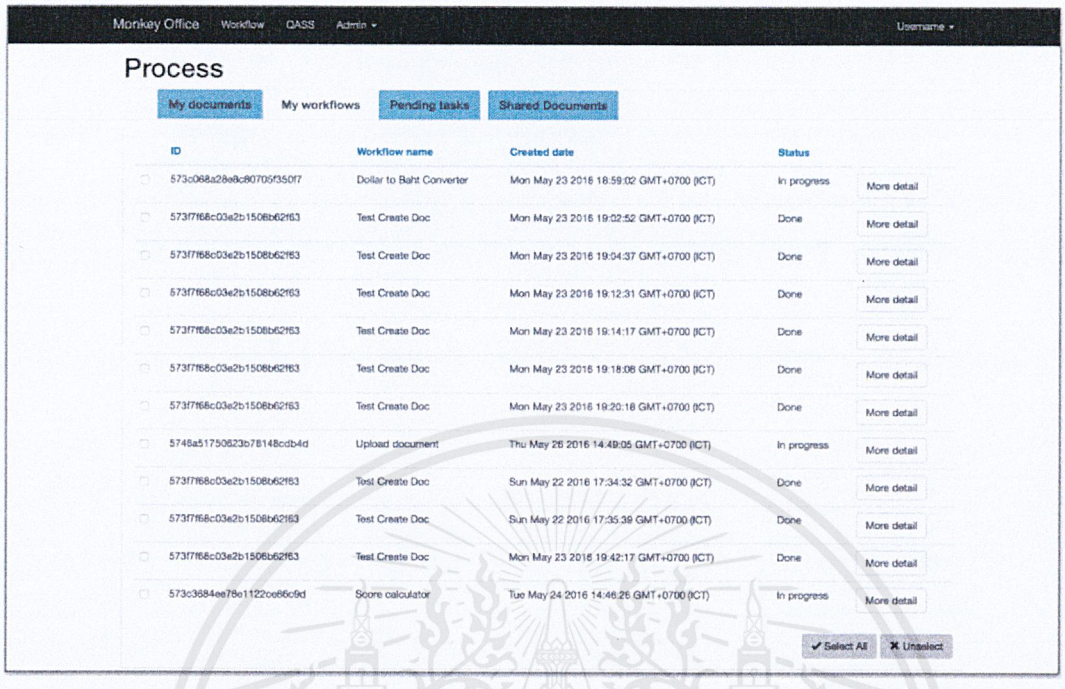

Figure 7.10. List of the workflow executions initiated by the current user

Figure 7.10 shows the list of the workflow executions initiated by the user that is currently logged in. The workflow executions that have not finised will have the status of "In progress", whereas those that have finished will have the status of "Done". Figure 7.11 shows the list of the tasks that await the user to perform.

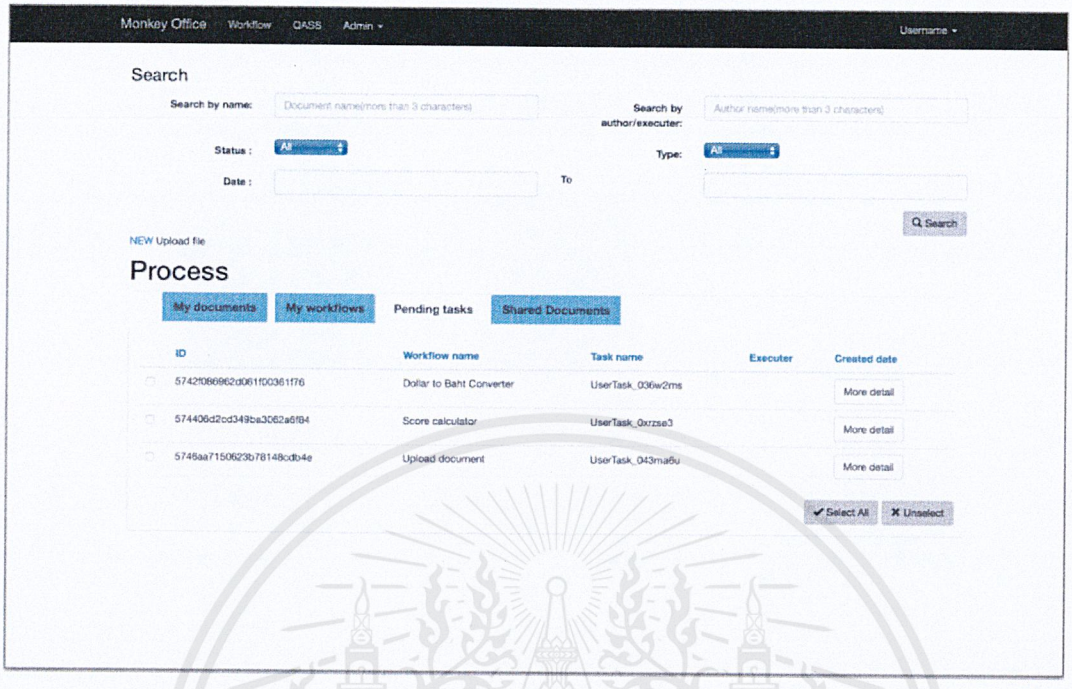

Figure 7.11. List of pending tasks

When the user wants to perform a task, the user just clicks on the button "More detail" for that task. The web form associated with that task will then be displayed for the user to perform the task.

Figure 7.12. Example of a form associated with a user task

Figure 7.12 shows the form associated with the user task for uploading a file. On this form, the user can enter the file name and then select a file on his/her computer to upload to the system.

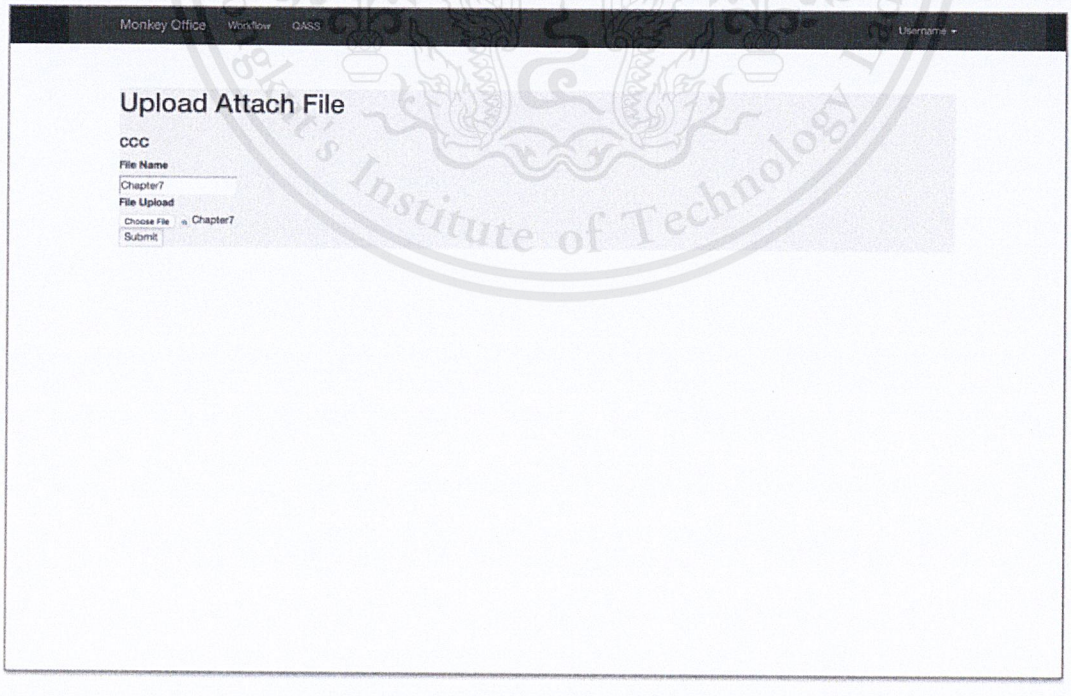

Figure 7.13. Form for uploading a file

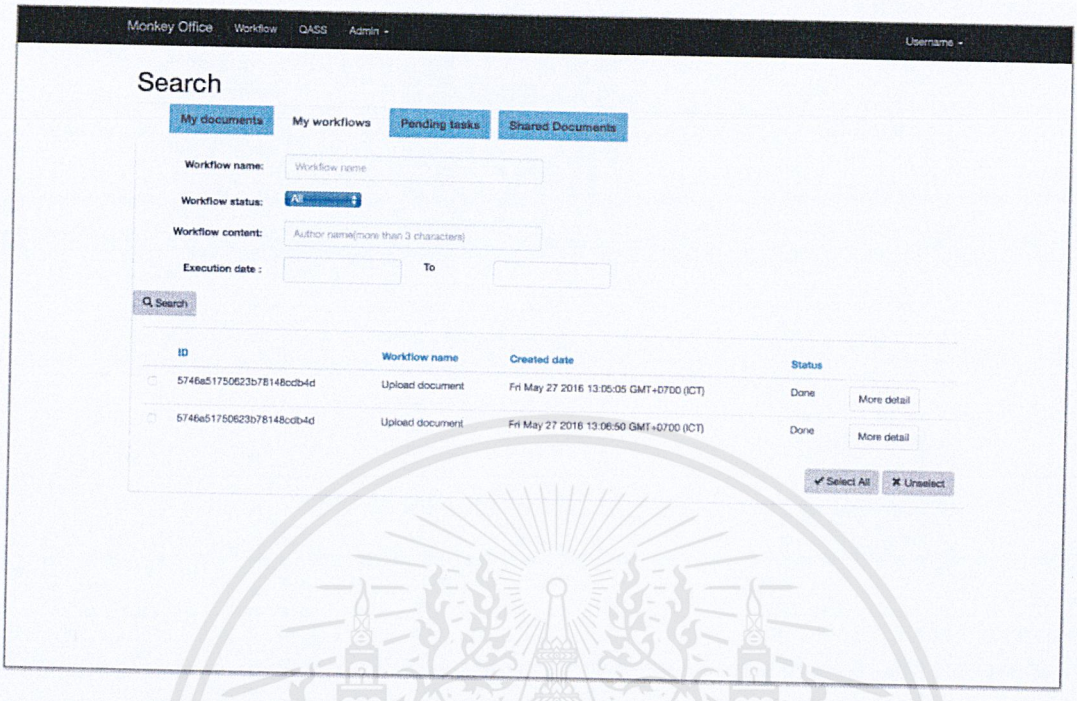

Figure 7.14. Page showing the status of each workflow execution

When we have finished the workflow execution, its status will be changed to "Done" and the document generated by the workflow and attached file will show in "My documents" tab.

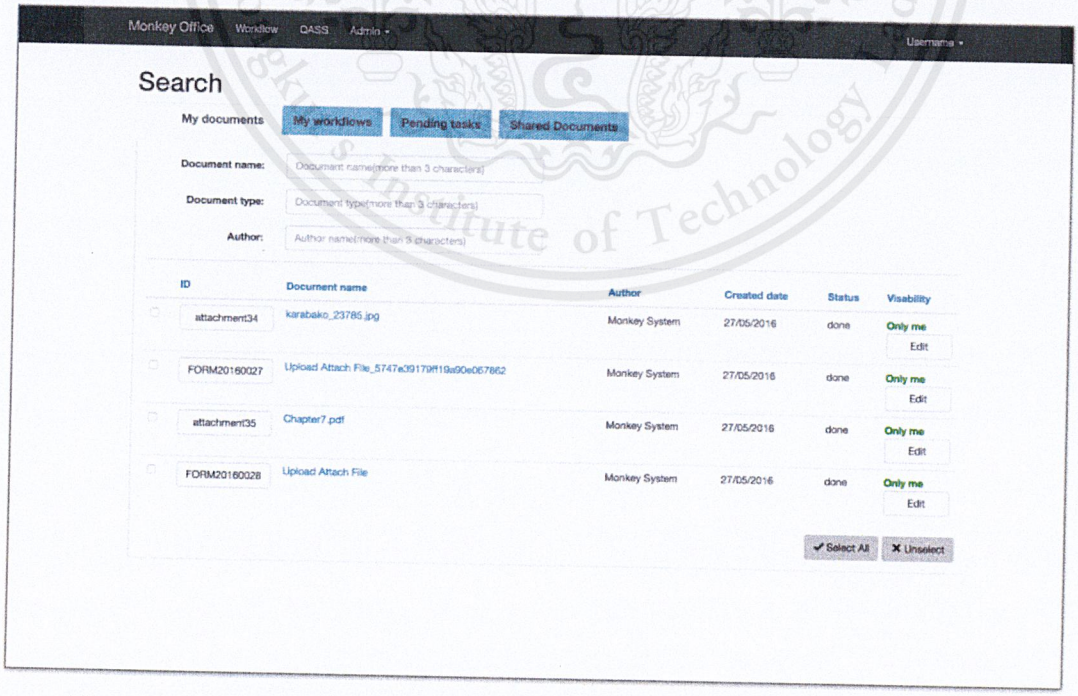

Figure 7.15. Page showing the list of the documents generated by a workflow execution or uploaded by the user during a workflow execution

Chapter 8
Conclusion

This chapter provides a conclusion of this project. The conclusion can be divided into three maia sections. First of all, Section 8.1 gives a summary of this project. Section 8.2, compare this project with existing systems. Finally, Section 8.3 talks about how our project can be developed in the future.

8.1 Project summary

We have developed the web-based workflow management system to support the International College, KMITL. This system has led the production of documents with the organization becomes more automated, more efficient, and less error-prone. Additionally, the managemrnt of workflows within the organization are computerized and can be tracked.

8.2 Comparison with other systems

8.2.1 Comparison with Bonita

Bonita is one of the most full feature workflow management system currently in the market. In fact, we adopt a number of ideas from Bonita. Being powerful, Bonita requires quite a steep learning curve. Our application can be seen as an attemp to simplify Bonita and make the system fully web-based.

8.2.2 Comparison with KiSSFLOW

KiSSFLOW is a very easy to use web-based workflow management system. However, it does not support BPMN notation. As a result complicated workflows are not supported. Also the user cannot create a custom task using scripts.

8.3 Future work

There are a number of possible improvements for our project.

1. Support advance BPMN v 2.0 elements and features such as loop activity, multi-instance activity.
2. Create commonly used service task types.
3. Support the use of data object and data store notation in BPMN diagram.

Bibliography

[1] Alrow, Jim, and Ila Neustadt. Introduction To BPMN 2: Non-Interactive Edition. Mountain Way Limited. Print.

[2] Wikipedia,. 'Business Process'. N.p., 2015. Web. 16 Oct. 2015.

[3] Appian,. 'Business Process Definition | Appian'. N.p., 2015. Web. 16 Oct. 2015.

[4] Wikipedia,. 'Business Process Model And Notation'. N.p., 2015. Web. 16 Oct. 2015.

[5] Silver, Bruce. *BPMN Method And Style*. 2nd ed. Aptos, CA: Cody-Cassidy Press, 2011. Print.

[6] Janelle Hill,. 'Do You Understand The Difference Between Workflow And BPM? - Janelle Hill'. N.p., 2010. Web. 18 Oct. 2015.

[7] Wikipedia,. 'Bonita BPM'. N.p., 2015. Web. 12 Oct. 2015.

[8] Bonitasoft.com,. 'Bonitasoft'. N.p., 2015. Web. 12 Oct. 2015.

[9] Groovy-lang.org,. 'The Groovy Programming Language'. N.p., 2015. Web. 12 Oct. 2015.

[10] Real-Life BPMN (2nd edition) by Jakob Freund, Bernd Rücker Camunda, 2014

[11] Baeyens,Tom.'Process Developments: Alfresco Creates Activiti'. *Processdevelopments.blogspot.com*. N.p., 2010. Web. 13 Oct. 2015.

[12] Alfresco.com,. 'Community | Alfresco'. N.p., 2015. Web. 13 Oct. 2015.

[13] Projects.spring.io,. 'Spring Integration'. N.p., 2015. Web. 14 Oct. 2015.

[14] Rademakers, Tijs. *Activiti In Action*. Shelter Island, NY: Manning, 2012. Print.