**Honda Virtual Assistant**

Project Workbook

**Members: Jason Xu, Jiankai Xu, Jeremy Lin**

**Table of Contents**

[Software Engineering Process 4](#_Toc58352377)

[Spider Diagram 4](#_Toc58352378)

[Reasoning 4](#_Toc58352379)

[Work Products by Phase 5](#_Toc58352380)

[Requirements 5](#_Toc58352381)

[Analysis 5](#_Toc58352382)

[Project Planning 5](#_Toc58352383)

[Task Tracking 6](#_Toc58352384)

[Requirements and Analysis 7](#_Toc58352385)

[Domain Analysis 7](#_Toc58352386)

[Problem Analysis 7](#_Toc58352387)

[Prioritized Requirements 7](#_Toc58352388)

[Identification 7](#_Toc58352389)

[Function 7](#_Toc58352390)

[Security 7](#_Toc58352391)

[Incident Report 8](#_Toc58352392)

[Use Cases 8](#_Toc58352393)

[User Searches Answers for Routine Tasks 8](#_Toc58352394)

[Inspector Looks Up Ticket on Incident 9](#_Toc58352395)

[Non-Functional Requirements 10](#_Toc58352396)

[Availability 10](#_Toc58352397)

[Performance 10](#_Toc58352398)

[Scalability 10](#_Toc58352399)

[Usability 10](#_Toc58352400)

[Testability 10](#_Toc58352401)

[Acceptance Plan 11](#_Toc58352402)

[Prioritized Requirement 11](#_Toc58352403)

[Non-functional Requirement 11](#_Toc58352404)

[Architecture 12](#_Toc58352405)

[Target Environment and System Architecture 12](#_Toc58352406)

[Inspector Login the System 13](#_Toc58352407)

[User Posts Queries 14](#_Toc58352408)

[System Identify Tasks 15](#_Toc58352409)

[User Posts Incident 15](#_Toc58352410)

[Subsystem Model 16](#_Toc58352411)

[Subsystem: Login System 16](#_Toc58352412)

[Subsystem: Bot Framework 16](#_Toc58352413)

[Subsystem: Routine Tasks Identifier 17](#_Toc58352414)

[Subsystem: Notification and Email Handler 18](#_Toc58352415)

[Subsystem: Incident Reporter 18](#_Toc58352416)

[Subsystem Interaction 19](#_Toc58352417)

[Login Process 19](#_Toc58352418)

[Query Process 20](#_Toc58352419)

[Incident Report 21](#_Toc58352420)

[Project Planning 22](#_Toc58352421)

[Project Schedule 22](#_Toc58352422)

[Iteration Plan for 1st Release 23](#_Toc58352423)

[Inception Phase: 23](#_Toc58352424)

[Elaboration Phase 24](#_Toc58352425)

[Iteration Plan for 2nd Release 24](#_Toc58352426)

[Risk Plan 24](#_Toc58352427)

[Project Handoff 27](#_Toc58352428)

[Login Subsystem, Notification & Email Handler 27](#_Toc58352429)

[Bot Framework & Routine Task Identifier 27](#_Toc58352430)

[Incident Reporter (ServiceNow) 27](#_Toc58352431)

[References 28](#_Toc58352432)

# Software Engineering Process

## Spider Diagram

*Figure 1: Spider Diagram for project.*

### Reasoning

Because of Personnel, Dynamism, and Criticality being closer to the center of the diagram shown above in Figure 1 and Table 1, we recommend an agile process that also takes some elements from a structured one to deal with Personnel being farther out in the diagram.

*Table 1: Explanation of Spider Diagram categories.*

|  |  |
| --- | --- |
| **Personnel** | Lowest.  Since all the team member doesn’t have much experience on the chat bot system before, except the sponsors, therefore the personnel are low. |
| **Dynamism** | Medium.  Given the communication with the sponsors, the changes/requirements per month is not extreme. But we noticed that multiple changes are required, therefore the scale is medium. |
| **Culture** | High.  Since the application is not very critical, the environment provided is free. Therefore, the scale is high. |
| **Size** | 5  Including sponsor and professor, only 5 members in our team. |
| **Criticality** | High.  Since this system thrives and evolves based on the defects, which will help the system to train their algorithm, therefore its criticality is high. |

## Work Products by Phase

The work products and a description of them we intend to use are outlined below in tables 2, 3, 4, and 5.

### Requirements

*Table 2: Work products for Requirements phase.*

|  |  |
| --- | --- |
| **Work Product** | **Description** |
| Prioritized Requirements | These describe what functionality the application should provide based on Use Cases and User Stories. |
| Use Cases | This section will provide list of actions that various roles can take in the system to achieve a goal |
| User Stories | This section will provide a list of features wanted by the various actors that partake in the software system |
| Non-Functional Requirements | These describe what functionality the application should provide based on Use Cases and User Stories. |
| Acceptance Plan | This work product will outline the tests that will be performed to see if the application is completed. The requirements will be agreed upon by the project manager, end user, and stakeholder for LinkedIn. |

### Analysis

*Table 3: Work products for analysis phase.*

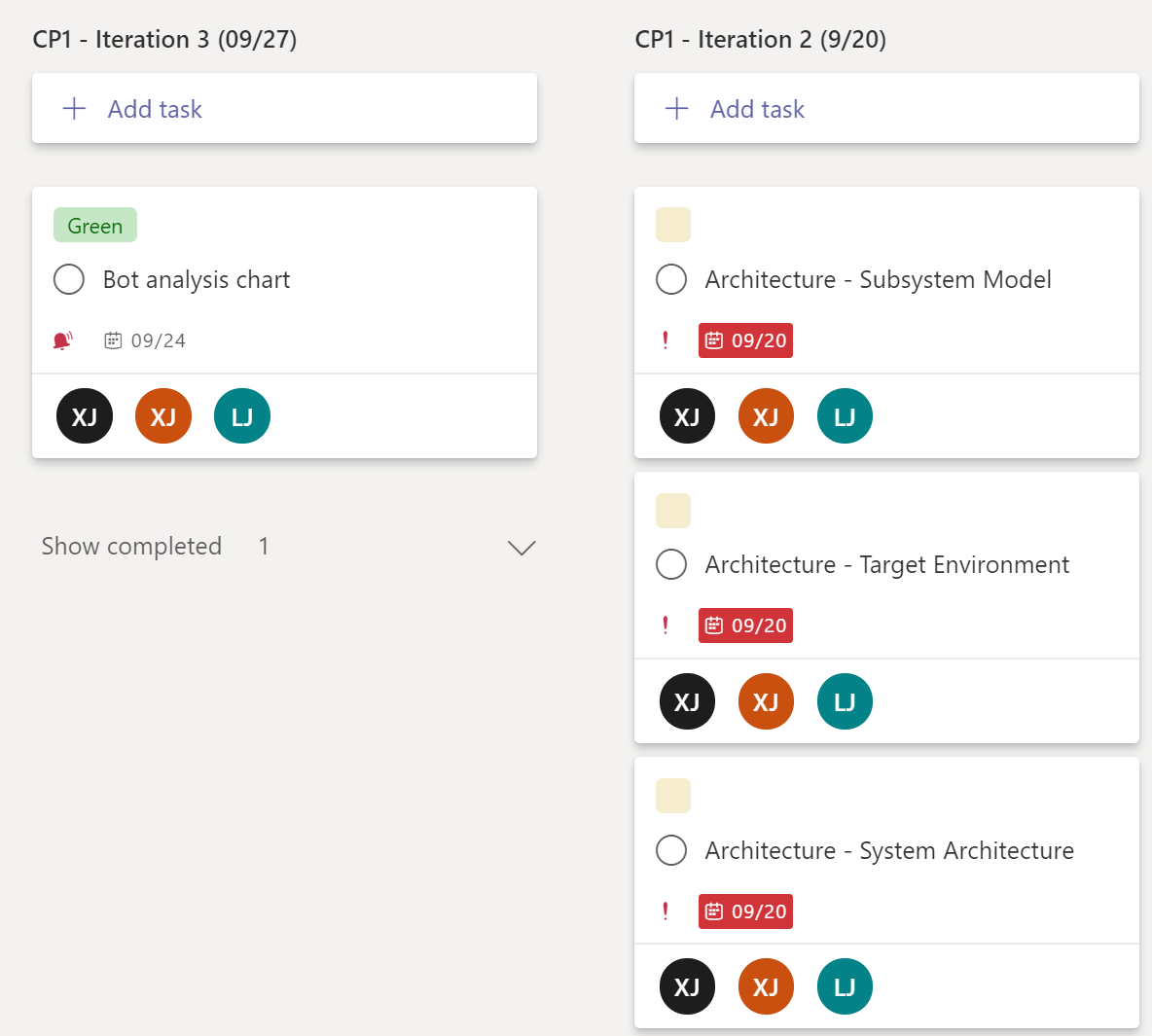
|  |  |
| --- | --- |
| **Work Product** | **Description** |
| Domain Analysis | This section will outline the analysis of the domain of this software feature and application. |
| Problem Analysis | This section will analyze the problem. |

### Project Planning

*Table 4: Work products for Project Planning phase.*

|  |  |
| --- | --- |
| **Work Product** | **Description** |
| Tentative Project Schedule | This will be a Gantt chart that shows the tasks needed to be performed, how long they took, and who was assigned to complete them. |
| Iteration Plan | The iteration plan outlines what happened during each iteration and is used to determine what the goal of each iteration should be |
| Risk Plan | The risk plan outlines any risks associated with the project. It will provide a description of the risk, how likely it is, and what solutions we can use to mitigate or eliminate the risk |

## Task Tracking



Assigned to

Due day

Tasks

Task size

Iterations

*Figure 2: Task Tracker for project.*

For task tracking, we used Microsoft Team task tracker tools. As shown in figure 2, tasks are divided based on their iterations. Currently our iteration span is one week, in the future planning, the end day of each iteration will be post at the end of the title bar.

In the task board, a label has been used to determine the task size – green means small, yellow means medium and red means large and beyond. Note that the due day and task content are described in the box. More detail information and comments are inside the task box. At the bottom of each task box, the people who is assigned to the tasks is marked.

# Requirements and Analysis

## Domain Analysis

Honda is the world largest motorcycle and internal combustion engines manufacturer. Honda’s product line also covers automobiles, electric generators, jet aircraft and robots etc. Honda provides reliable services as a multi-industry company. As an international company, Honda developed a knowledge base for employees. In the digital field of the company, many tickets have been sent during the working hours, relating to trivial questions. However, it’s only been with the IT helpdesk that we have seen a 60% of tickets raised being responded to. More efficiency is required into this internal helpdesk channel.

## Problem Analysis

As the largest motorcycle manufacturer since 1959, this giant machine has hired thousands of employees. (Grant and Neupert, 2003) With this huge number of employees, finding a better internal user support is significant not only contributing to improve the efficiency, but also help company save costs. Given that the challenge of geographical constraints, removal work support, new and complex engineering applications, skill building of existing resources and tediousness of routine tasks, etc, 80% of employees want self-service tools to be more productive in the digital workplace. (Govindwar A., 2020) The idea of building a high-level virtual assistant has been brought to the counter. Two features are all-important for this virtual assistant: provide an intelligent self-service kiosk and chat-bot to handle support requirements.

## Prioritized Requirements

### Identification

1. The system MUST be able to identify routine tasks based on historical data.

### Function

1. The system MUST be able to recognize the service user requires and provide related documents linked to knowledge base or response accordingly, with proper searching abilities.
2. The interface MUST integrate with existing interface (skype and Microsoft team).

### Security

1. The user CAN only access the system with the authorized domain network environment.
2. Any personal information and dossier involved between user and bots MUST be private and secluded from other users.

### Incident Report

1. The system MUST be able to open an incident ticket if the services cannot be provided.
2. The system MUST track tickets reported, log metadata and analyze incidents type and frequency.
3. The system MUST allow inspector to access the data for analysis and maintenance purpose.

## Use Cases

*Table 5: Descriptions of actors in system.*

|  |  |
| --- | --- |
| **Actor** | **Description** |
| User | An external user who will use the system to finish their routine tasks with self-service chatbot. |
| Inspector (IT) | An external user of the system who will examine the ticket and analysis of incidents. They are responsible of handling these incidents. |

### User Searches Answers for Routine Tasks

1. **Functional Requirements:**  1, 2, 3, 4, 5, 6
2. **Description:** User wants to find the answers to some routine tasks he/she is on. He/She turns to the system for some quick and accurate answer of how to solve it.
3. **Scenarios**

*Table 6: Scenarios for User Searches Answers for Routine Tasks use case.*

|  |  |
| --- | --- |
| **Actor** | User |
| **Assumptions** | * User is in authorized network environment |
| **Ideal Path** | 1. User posts his tasks into system. 2. System recognizes the tasks as routine tasks. 3. System search through existing knowledge base and give response to user of how to solve the tasks. 4. System waits for response from user to see if the answer solves the tasks. 5. User says yes, go to step 6. User says no, go to step 1. 6. System ends the conversation. |
| **Other Paths** | * The tasks are not recognized as routine tasks. System records the data for identifying routine tasks. * After 3 cycles, the user is not admitted the answers provided. An incident ticket is reported, prompting user that manual help is on the way. |
| **Error Paths** | * The tasks user posts in not in the knowledge base. An incident ticket will also be reported for manual help. * The user is accessing the system outside of the authorized domain network, the request will be declined. |
| **Outcomes** | * User gets the answer for tasks. |

1. **User Stories**

Table 7: User stories for User Searches Answers for Routine Tasks.

|  |  |  |
| --- | --- | --- |
| **As a...** | **I want to...** | **So that...** |
| User | Look up some files in the knowledge base. | I can finish my job which requires certain documentation. |
| User | Find out some answers for system operating. | I can use the system well to increase my efficiency. |
| User | Get the answers of daily routine tasks. | I can finish my routine tasks more efficient. |
| User | Keep my query with system private. | Others won’t access my privacy. |

### Inspector Looks Up Ticket on Incident

1. **Functional Requirements:** 4, 6, 7, 8
2. **Description:** An inspector sees an incident reported by ticket. He/she looks up the metadata surrounding the incident to investigate the issue.
3. **Scenarios**

*Table 8: Scenarios for Inspector Looks Up Ticket on Incident use case.*

|  |  |
| --- | --- |
| **Actor** | Inspector |
| **Assumptions** | * Inspector is in authorized network environment |
| **Ideal Path** | 1. Inspector gets the notifications of new incident reported. 2. Inspector looks up the ticket, if the incident is based on semantics, go to step 3, else go to 5. 3. Inspector responses the incident. 4. Inspector collects the data, and he/she tunes the system based on the data for system performance improvement, go to step 6. 5. If it is not based on semantics, the inspector should request a maintenance from technics, go to step 6. 6. Inspector ends inspection session. |
| **Other Paths** | * Even inspector cannot understand the semantics error, he/she should resort to the user for the meaning of the ticket, recording the data to improve the system. * Inspector’s response is not helpful, a meeting based on the incident needs to be held to solve the problem. |
| **Error Paths** | * The inspector gives the wrong response, causing user to fail their tasks. |
| **Outcomes** | * Inspector fixes the incident. |

1. **User Stories**

Table 9: User stories for Inspector Looks Up Ticket on Incident.

|  |  |  |
| --- | --- | --- |
| **As a...** | **I want to...** | **So that...** |
| Inspector | Fix the incident. | User can finish their query with the system. |
| Inspector | Upload new knowledge store files | The system has better coverage of stored knowledge. |
| Inspector | Analyze why the incident is reported. | Improve the performance of the system. |
| Inspector | Make sure myself doesn’t make mistake on my responses. | I won’t misguide the user. |

## Non-Functional Requirements

### Availability

* The system should be available 24/7 except for monthly maintenance. If the system is not working, an alarm should be sent to responsible technic support faculty.

### Performance

* No first response time should be longer than 5 sec.
* The bot should be able to handle nature language processing.
* The bot interact should act like human.

### Scalability

* System should remain responsive when it is occupied by 1000 users at the time.

### Usability

* This system will be able to run on multiple devices such as computer, tablets, and phones.
* The system should be user friendly to user, maintainer and inspector.

### Testability

* The system will undergo unit testing and integration tests for individual components.
* End-to-end tests will be done to ensure a user can complete the service.

## Acceptance Plan

### Prioritized Requirement

* All prioritized requirements must pass testing done by the stakeholders corresponding to each one.
* Each user story should be run successfully by the stakeholders.

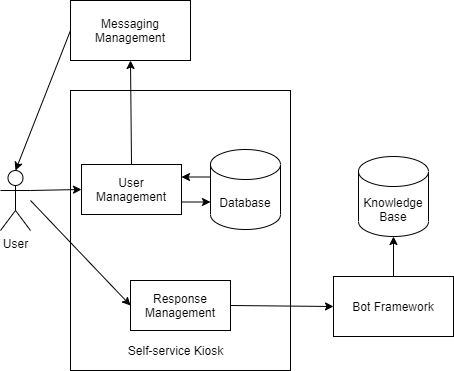
### Non-functional Requirement

* Availability testing: A repeated query test system will be set up for a 24-hour cycle to see if they will get correct response throughout the test.
* Performance testing: A user/tester will need to complete the user story scenarios mentioned in the use cases, recording time spending and accessing the language nature of response. The expecting responsive time is 5 sec. Any time exceeds the threshold will be taken as a fail.
* Scalability testing: The system should handle 1000 users querying at the same time, to replicate this, our testing will send out same amount of message at the same time to see whether the system will collapse due to this traffic.
* Usability test:
  + This system will be able to run on multiple devices such as computer, tablets, and phones, the bot will be tested not only on desktops, but also other mobile devices. The mainstream operating system (Windows, macOS, Linux) should be able to run the system.
  + The system should be user friendly to user, maintainer and inspector. The user and shareholder will be asked to access the UI layout and other environment features, returning feedbacks for future tuning.

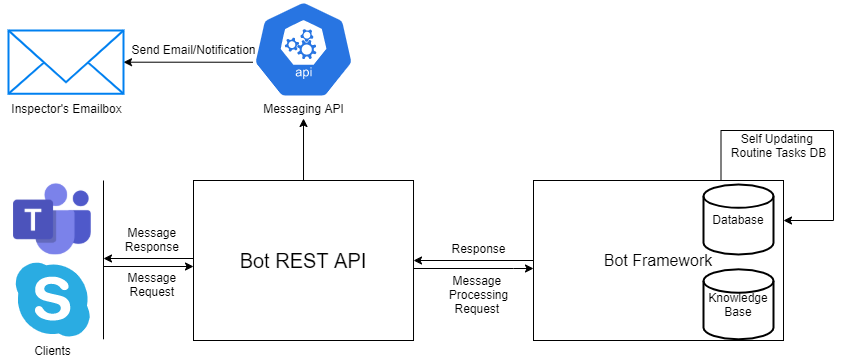
# Architecture

## Target Environment and System Architecture

Note that there are two possible channels for user, one is Microsoft Teams and the other one is Skype. The Bot REST API will be written in node.js and deployed to Azure and it will serve as an interface to Microsoft teams, which is primarily how the end user interacts will the knowledge base. The Bot REST API will connect to chosen bot framework, which will include the knowledge base and database. The bot framework includes the basic function of a chatbot. However, extra function needs to be created, since a database and knowledge base is involved. The chatbot will need to search for the answer within the knowledge base. The database will be used to store admin credentials and ticketing metadata. The database solution we intend to use is MongoDB because of its scalability and offering of cloud products. The Messaging API service will be provided by Twilio. It will be used as a channel to provide access code to inspector, as well as sending notification to responsible inspector once a ticket has been issued.

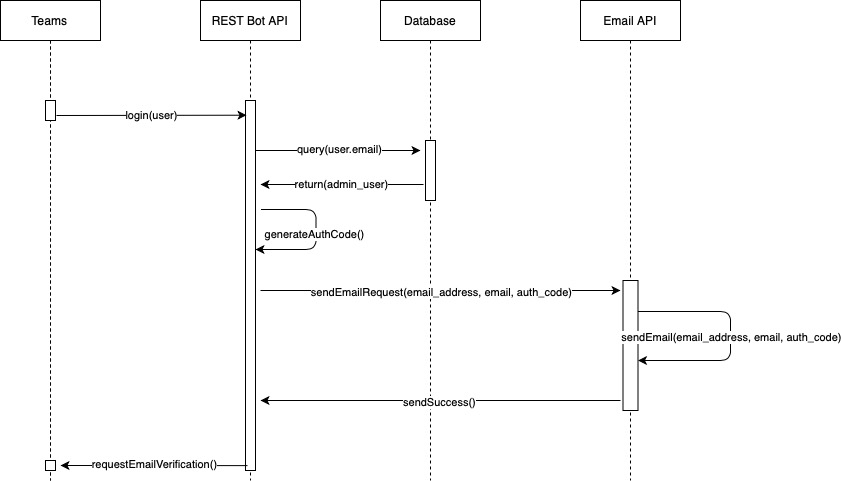


*Figure 3: Conceptual Architecture*



*Figure 4: System Architecture*

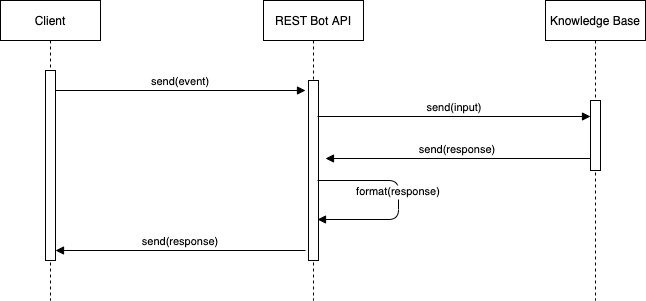
### Inspector Login the System



*Figure 5: Sequence Diagram of User/Inspector Login the System*

The user/inspector login the system, they are required to provide email address and password to the system. The system will first match the data with database. If succeed, an access code is generated, sent to email address provided for verification. If the verification is passed, the user/inspector can access the system.

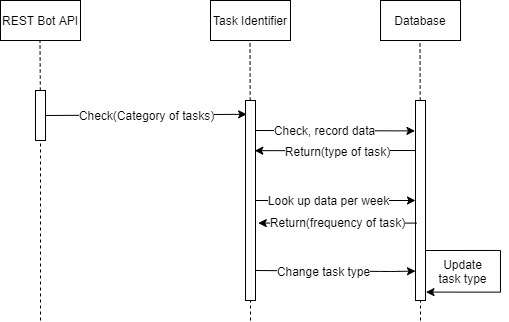
### User Posts Queries



*Figure 6: Sequence Diagram of User Posts Queries*

The user/inspector will post their queries to the Bot REST API, the API will request response from our bot framework with given knowledge base. With the proper answers, the Bot REST API will format the response and provide multiple options (if applicable) for user to choose from.

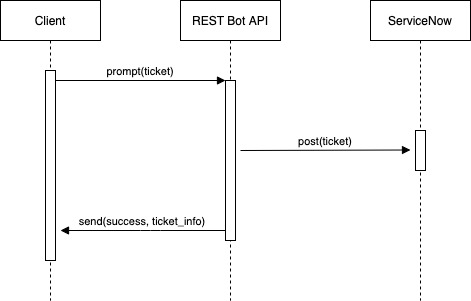
### System Identify Tasks



*Figure 7: Sequence Diagram of System Identify Tasks*

The task identifier will look up the data in database, about its task type. A routine task’s response will be store in system instead of looking up the whole knowledge base for efficiency. The system will also update the tasks in database based on its frequency.

### User Posts Incident



*Figure 8: Sequence Diagram of User Posts Incident*

When user is not satisfied with the options provided by the system, he/she will be able to posts a ticket. The system will post the ticket using the ServiceNow to prompt responsible inspectors for the issue.

## Subsystem Model

### Subsystem: Login System

This system provides the functionality related to login process. Since the system is embedded into Microsoft Teams or Skype, the system will check user’s credential when first interact with the system. If the account matches the active user list in the database, the system will allow that user logging in. For inspector, however, the message API will send a verification email to the email address restored in the database. The inspector needs to provide verification code to login as an inspector. Our messaging API will be responsible of sending those emails.

**Prioritized requirements:**

3, 4

**Non-functional requirements:**

* Availability: The login process should be available 24/7 except for monthly maintenance.
* Performance: The response time should not be longer than 5 sec.
* Scalability: The login process should be working concurrently with 1000 users.
* Usability: The login system should be working on multiple devices and mainstream operating systems (Windows, macOS, Linux)

**Contracts**:

* Provide Access Code – The system will provide access code for inspector, making sure the code is sent to the email address that is stored in the database.
* Identity Verification – The system will check user’s credential they are using through the channel (Microsoft Teams or Skype). If the account matches the active user list, the login process can proceed.
* Update Active User List – The system will provide inspector the access to the active user list. The inspector will be able to modify the list in order to give someone access or removing their access.

### Subsystem: Bot Framework

This subsystem will process the message send from user, retrieve the keyword of the questions, and figure out user intent and provide related response.

**Prioritized requirements:**

2, 3, 4

**Non-functional requirements:**

* Availability: The login process should be available 24/7 except for monthly maintenance.
* Performance: The response time should not be longer than 5 sec; The key word retrieval should be accurate.
* Scalability: The processor should be working concurrently with 1000 users.
* Usability: The login system should be working on multiple devices and mainstream operating systems (Windows, macOS, Linux)

**Contracts:**

* Retrieve Keyword – The system will retrieve the keyword from user’s message. Those keywords should be used for determining user intent.
* Provide Response – The system will provide response based on the user intent and knowledge base data. In addition, the user intent response can be multiple possible options.

### Subsystem: Routine Tasks Identifier

This system would be able to identify the top routine tasks based on the interaction with the users in a certain period (2 weeks). The primary intents of the users would be stored in a database and analysis by the system to determine the top routine tasks. The result would be posted every two weeks, and the collection system would refresh its records.

**Prioritized requirements:**

1

**Non-functional requirements:**

* Availability: The subsystem should be available 24/7 except for monthly maintenance.
* Scalability: The subsystem should be working concurrently with 1000 users.
* Usability: The subsystem should be working on multiple devices and mainstream operating systems (Windows, macOS, Linux)

**Contracts:**

* Identify Task – The system should be able to identify the task from the message user provided.
* Update category of task – With data recorded based on user queries, the subsystem will update its database of routine tasks. If the frequency of some tasks exceeds the threshold, the system will update the tasks category.

### Subsystem: Notification and Email Handler

This subsystem will implement the function of receiving and sending messages. Cooperating with login system, the handler will send out verification letter towards the provided address. With incident report system, the handler will send out notification to inspector, apprising them of newly received incident ticket.

**Prioritized requirements:**

6, 7, 8

**Non-functional requirements:**

* Availability: The subsystem should be available 24/7 except for monthly maintenance.
* Scalability: The subsystem should be working concurrently with 1000 users.
* Usability: The subsystem should be working on multiple devices and mainstream operating systems (Windows, macOS, Linux)

**Contracts:**

* Send Verification Email – The system will send verification email to provided email address with access code associated with their role.
* Notify Inspector – The system will notify responsible inspector of new incident tickets incoming.

### Subsystem: Incident Reporter

This subsystem will generate an incident ticket if the information processor’s response does not meet user’s requirement. Cooperating with message handler, the notification will be sent to responsible inspector. The ticket data will be stored in databases, including bot’s responses and the keyword retrieve by the processor. An incident analysis report will be generated per week, focusing on the type, the frequency, the user of the incident, etc. The analysis should be able to improve the system.

**Prioritized requirements:**

6, 7, 8

**Non-functional requirements:**

* Availability: The subsystem should be available 24/7 except for monthly maintenance.
* Scalability: The subsystem should be working concurrently with 1000 users.
* Usability: The subsystem should be working on multiple devices and mainstream operating systems (Windows, macOS, Linux); The subsystem should be user friendly.

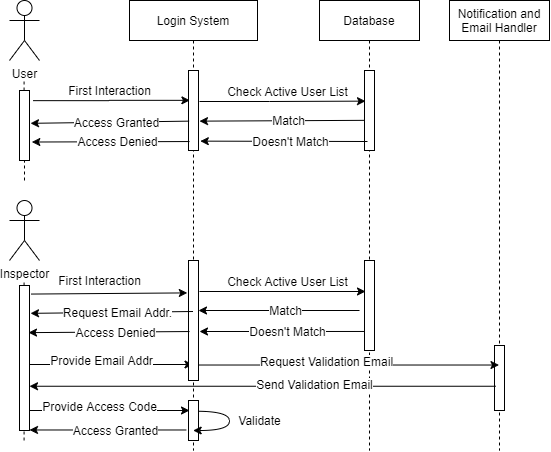
**Contracts:**

* Generate Incident Ticket – The incident ticket should be generated if the response is not accepted by the user. The ticket should be stored into two forms: a notification will be sent to responsible inspector; the local data storing the ticket information for inspector to process once they login.
* Generate Incident Analysis – An analysis will be generated per week, based on the type, the frequency, the user of the incident, etc. For example, question like the quantity of remaining request, the quantity of unsolvable request or solvable request, the keyword of the unsolvable request and the type of ticket that can be automated should be takin into consideration of this system.

## Subsystem Interaction

To demonstrate the interactions between users and the subsystems, sequence diagrams encapsulating subsystem interactions have been created.

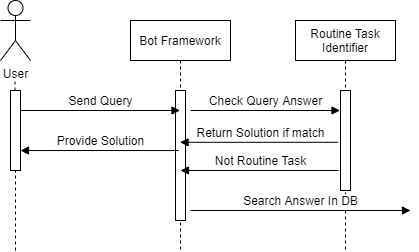
### Login Process



*Figure 9: Sequence Diagram of Login Process*

The process involves the login subsystem as well as the notification and email handler. The notification and email handler will only be involved when the protagonist is inspector. An email with access code will be sent to provided address, if the address matches active user list. The inspector will need to valid their access code to have the access to the system.

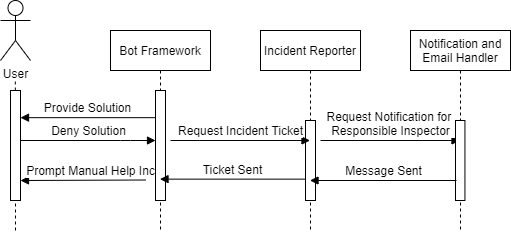
### Query Process



*Figure 10: Sequence Diagram of Query Process*

The process involves Bot Framework, Routine Task Identifier and Database. When user sends out the query, the bot framework will first determine user intent, and check if related solution is in routine task identifier. If matches the records, return the answer directly. If not, the bot framework will search through database instead.

### Incident Report



*Figure 11: Sequence Diagram of Incident Report*

The process involves bot framework, incident reporter and notification and email handler. When answer provided is not accepted by user, the bot framework will request the incident reporter to issue a ticket, including all the information and type of ticket. The incident reporter will request the notification and email handler to send an email to current responsible inspector. Once the email has been sent, the incident reporter will let bot framework know, and the bot framework will prompt the user that manual help is on the way.

# Project Planning

## Project Schedule

*Table 10: Start and end dates for components of project.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Activities** | **Start Date** | **End Date** | **Duration** |
| **Project Preparation** |  |  |  |
| Domain & Problem Analysis | 9/7/2020 | 9/8/2020 | 1 |
| Requirements Draft | 9/7/2020 | 9/10/2020 | 3 |
| Requirements Analysis | 9/10/2020 | 9/11/2020 | 1 |
| Prioritize Requirements | 9/10/2020 | 9/14/2020 | 4 |
| Research on Bot Framework | 9/7/2020 | 9/13/2020 | 6 |
| Project Schedule Estimation | 9/14/2020 | 9/17/2020 | 3 |
| Project Risk Plan | 9/14/2020 | 9/15/2020 | 1 |
| Use Cases | 9/14/2020 | 9/21/2020 | 7 |
| Acceptance Plan | 9/14/2020 | 9/15/2020 | 1 |
| **Bot Framework Report** |  |  |  |
| Pros & Cons Report | 9/13/2020 | 9/20/2020 | 7 |
| Bot Analysis Chart | 9/21/2020 | 9/24/2020 | 3 |
| Report Selection Result | 9/24/2020 | 10/1/2020 | 7 |
| **Bot Framework Demonstration** |  |  |  |
| Bot Demo1 Construction | 9/27/2020 | 10/21/2020 | 24 |
| Bot Demo2 Construction | 10/21/2020 | 10/28/2020 | 7 |
| Demo to sponsors with decision making (analysis table) | 10/28/2020 | 10/29/2020 | 1 |
| **Use Cases: User Searches Answers for Routine Tasks** | 10/29/2020 | 11/26/2020 | 28 |
| **Use Cases: Inspector Looks Up Ticket on Incidents** | 11/26/2020 | 12/6/2020 | 10 |
| **Testing** |  |  |  |
| Acceptance Plan Testing | 12/6/2020 | 12/7/2020 | 1 |
| Unit Testing | 12/6/2020 | 12/7/2020 | 1 |
| **Evaluation Report**  Bot Framework Analysis; Instructions of installation; Ticket Analysis Evaluation | 12/1/2020 | 12/7/2020 | 7 |
| **Milestone: Release 1** |  | 10/28/2020 |  |
| **Milestone: Release 2** |  | 12/8/2020 |  |

*Chart 1: Start and end dates for components of project.*

Release 2

Release 1

## Iteration Plan for 1st Release

### Inception Phase:

Iteration 1:

* Meeting with sponsor
* Domain analysis, requirements prioritizing requirements and non-functional requirements
* Research on bot framework

Iteration 2:

* Collect Bot framework candidates
* Analyze bot framework candidates according to selected factors
* Analyze and plan for potential project risks
* Come up with uses cases, user stories and acceptance plan

Iteration 3:

* Make tentative project schedule, iteration plans
* Modify use cases, acceptance plan based on the requirements
* Develop bot analysis chart with Honda’s decision-making criteria
* Report current work product and progress

### Elaboration Phase

Iteration 1:

* Draft report for bot framework analysis
* Meeting with sponsors about any modifications

Iteration 2:

* Building Demo for Azure bot
* Building Demo for Google Dialogflow

Iteration 3:

* Demonstrate bot framework
* Compare the performance of each demo.
* Finish report of bot framework analysis

## Iteration Plan for 2nd Release

Iteration 1:

* Use Cases: User Searches Answers for Routine Tasks
* Goal: Be able to response with accurate answers to the user on the intended routine tasks
  + Develop algorithms for selecting routine tasks
  + Develop architecture to response based on the knowledge base information
  + Test

Iteration 2:

* Use Cases: Inspector Looks Up Ticket on Incidents
* Goal: System be able to allow inspector to look up incident tickets
  + Create inspector interface for ticket reporting
  + Develop architecture for ticket data analysis
  + Test

Iteration 3:

* Test current work products based on acceptance plan and fulfill the non-functional requirements
* Finish a report on work products
* Demonstrate customized bot framework

## Risk Plan

*Table 11: Overview of schedule risk.*

|  |  |
| --- | --- |
| Risk or Issue | Project falls behind schedule |
| Brief Description | Project must be delayed because of missing functionality, group members unfamiliarity with technology, or other factors that can possibly lead to project being cancelled |
| Criticality (H|M|L) | M |
| Probability of risk being realized | H |
| Mitigation Strategy (Avoid | Mitigate | Contain) | Contain |
| Action Item | Use scrum to check progress. Use incremental design. Prioritize functionality. Start early. |
| Assigned to (role name) | Project Manager |

*Table 12: Overview of working environment risk*

|  |  |
| --- | --- |
| Risk or Issue | Team is unfamiliar with working environment |
| Brief Description | The programming language and working environment may be new to some team members, adding up the cost of learning and adapting the new environment |
| Criticality (H|M|L) | H |
| Probability of risk being realized | H |
| Mitigation Strategy (Avoid | Mitigate | Contain) | Mitigate |
| Action Item | Research tools and environment in project planning. Pair programming with people that have knowledge of the environment. |
| Assigned to (role name) | Project Manager |

*Table 13: Overview of working environment risk*

|  |  |
| --- | --- |
| Risk or Issue | Poor team dynamic |
| Brief Description | On project planning, team members argue over how to complete tasks and progress with project. |
| Criticality (H|M|L) | L |
| Probability of risk being realized | H |
| Mitigation Strategy (Avoid | Mitigate | Contain) | Mitigate |
| Action Item | Majority rule decides progression and method of task completion. Consulting sponsors for ideas and mediation. |
| Assigned to (role name) | Project Manager |

*Table 14: Overview of requirement creep risk*

|  |  |
| --- | --- |
| Risk or Issue | Requirement creep |
| Brief Description | Scope or requirements grow after the project has started and developers become overwhelmed trying to handle it. |
| Criticality (H|M|L) | H |
| Probability of risk being realized | L |
| Mitigation Strategy (Avoid | Mitigate | Contain) | Mitigate |
| Action Item | Meet with stakeholders and end users before project starts to collect requirements and meet often. Identify multiple use cases. Prioritize the requirements. Create clear acceptance plan for non-functional requirements. |
| Assigned to (role name) | Project Manager |

# Project Handoff

Since this project is comprised of three parts: searching bot framework, making decision analysis about bot framework, and creating bot framework demo, we only have one month to work on building the system. Therefore, our team have not finished building all the subsystem described above. However, we indeed have made some progress. Since this project is going to be carried on next semester, a handoff has been presented for future reference if possible.

## Login Subsystem, Notification & Email Handler

The login system and notification & email handler has not been built into our system. Our idea of the login system is that the system itself will have a database for storing credential. When the user first interacts with the system, the system will ask him if he wants to sign in as an inspector. If true, the system will check the database, and sending the email with access code to the email address already recorded in the database. The user needs to provide access code to have access to interact as a role of inspector.

For the notification & email handler part, we intended to make use of Twilio.io. Its messaging API should be able to handle this job.

## Bot Framework & Routine Task Identifier

Since our bot framework is built on LUIS. It requires large amount of training data including intents and entities. For us, we do not have time to feed the training data into the system, but we did create an intent named “go\_cognitive”, which will lead the bot framework to the knowledge base file. Other intents should be provided for other uses. For example, daily conversation, work reminders, etc.

In addition, since we were not provided with existing knowledge base file, we feed the cognitive search service with .doc file we created. If the static web page or other knowledge base file is provided in the future, using parser or [web crawler](https://github.com/thomas11/AzureSearchCrawler) would be efficient.

Our idea of routine task identifier is that the system should be able to upload intents & utterance programmatically from local. This [link](https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-tutorial-node-import-utterances-csv) provides the code and \*.csv file which can be parsed to create a NEW LUIS App. However, we have not figured out how to update the existing LUIS App yet. Once this can be done, the routine task identifier should collect information from user every time they provide an answer to the user. If the user is not accepting the response, the system should either create a new utterance based on the incident reporter or add new utterance into existing intents. The task that has been querying frequently will fall into the “routine task” category.

## Incident Reporter (ServiceNow)

We have set up a ServiceNow instance to collect reported incidents and for the administrative teams to review incidents that were reported through the Honda Virtual Assistant. A script for running a POST request with the incident information has already been written and an intent in LUIS has been made to identify user inputs that suggest wanting to make a ticket request.

We, however, have not made a Bot Framework dialog workflow for the user to input incident information. In the future, a new child dialog should be developed to collect incident information by an adaptive card and send that incident information using the POST request script that we developed.

# References

Grant, Robert M.; Neupert, Kent E. (2003). *Cases in contemporary strategy analysis (3rd ed.)*. Wiley-Blackwell. ISBN 1-4051-1180-1.

Govindwar, A. (2020, Sep 3). *Virtual Assistant Research*. Honda.

OSU CSE Department (Unknown). *Analysis, Requirements, and Processes for a Price Prediction Model for Airbnb*. Ohio State University. URL: <https://osu.instructure.com/courses/90553/files?preview=22881110>

Xu J.; Chatla, S.; Pifer Z.; Nadkarni S. (2020). *LinkedIn Course System for Lecturers and Users.* Ohio State University. URL: [https://docs.google.com/document/d/1vKoGokqzAOiAyfQPPbZzxdTEri0XWCPd/edit#](https://docs.google.com/document/d/1vKoGokqzAOiAyfQPPbZzxdTEri0XWCPd/edit)