A GITHUB-BASED VOICE ASSISTANT FOR SOFTWARE DEVELOPERS AND TEAMS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Siriwan Sereesathien

June 2021

COMMITTEE MEMBERSHIP

TITLE:	A GitHub-based Voice Assistant for Software Developers and Teams
AUTHOR:	Siriwan Sereesathien
DATE SUBMITTED:	June 2021
COMMITTEE CHAIR:	Bruno da Silva, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Franz J. Kurfess, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Nasir Eisty, Ph.D. Professor of Computer Science

ABSTRACT

A GitHub-based Voice Assistant for Software Developers and Teams

Siriwan Sereesathien

Software developers and teams typically rely on source code and tasks management tools for their projects. They tend to depend on different platforms such as GitHub, Azure Devops, Bitbucket, and GitLab for task-tracking, feature-tracking, and bug-tracking to develop and maintain their software repositories. Individually, developers may lose concentration when having to navigate through numerous screens consisting of various platforms to perform daily tasks. Additionally, while in meetings (non-virtual), teams are often separate from their machines and often would have to rely on pure recollection of the tasks and issues related to their work. This can delay the decision-making process and take away valuable focus hours of developers. Although there is usually one person with their laptop to guide the meeting and has access to the source code management tools, this can take a lot of time as they are not familiar with all the developers' independent works. Therefore, a new tool needs to be introduced to help accelerate individual and team meetings' productivity. In this paper, we continued the work on Robin, a voice-assistant built to answer questions regarding GitHub issues and source code management. Robin has the ability to answer questions in addition to completing actions on the behalf of the developer. This thesis presents Robin's abilities, architecture, and implementation while also examining its usability through a user study. Our study suggests that some people love the idea of having a conversational agent for software development. However, a lot more research and iterations must be done to fully make Robin give the user experience we imagined. In this thesis, we were able to set the foundation of this idea and the lessons that we learned.

ACKNOWLEDGMENTS

I would like to acknowledge and thank my thesis advisor, Dr. Bruno da Silva, for the entire year of his guidance and advice for this thesis. Thank you for also being my software engineering and capstone professor for the last two years. I learned a lot from you as a student and as a professional. In addition, I would like to extend my gratitude to my committee members, Dr. Franz Kurfess and Dr. Nasir Eisty, for spending the time and energy in reviewing my thesis. Thank you Dr. Kurfess for also being my human computer interaction as well as artificial intelligence professor here at Cal Poly.

I would also like to thank my friends and classmates who have always been by my side. Thank you for both helping me and struggling with me through these incredible 4 years of college.

I also would like to thank my family for their constant support and understanding. If it weren't for them, I would not have chosen the field of computer science and be where I am today. They taught me how to work hard and never give up on whatever I do. Thank you for always being there when I need advice and help. Also, many thanks to my sister for taking care of me and providing for me in all our fun foodie tours and adventures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Software development is not an easy endeavor. Studies have shown that developers often have to perform a variety of fragmented tasks, constantly having to switch applications and contexts [8] [10]. Developers have a series of low-level workflows that they must combine together in order to complete a high level task [5]; creating pull requests, making sure the tests passed, notifying their reviewing team, adding comments, and then ultimately merging the pull request and closing the related issue. Constantly having to keep a mental model of source code, version control, issues, tasks, tests, and discussion threads adds a lot of cognitive load and stress to a software engineer [9]. These alongside other cumbersome daily tasks such as planned meetings, checking emails, and other inherent interruptions that go on during a work day often leads to a decrease in developer productivity [10]. It would be beneficial if all these small, fragmented tasks could be assisted or partially automated by tools.

A Stack Overflow Developer Survey [1] shows that "meetings", second to "distracting work environment", are what developers believe to be the greatest challenges to productivity. The daily stand-up meeting (DSM) is one of the most common agile practices today, as concluded in [11]. However, these intended 15-minute meetings often go overtime. Although just a few minutes may not seem like much, having this happen daily ultimately adds up to take away valuable productive hours of developers. With overtime meetings in addition to scattered/fragmented tasks, software teams often struggle to achieve optimum productivity.

We aimed to help software teams increase productivity by automating repetitive, fragmented tasks that are typical in software developers' workflow, in and out of meetings. Here, we implemented Robin, an assistant that accepts issues and source code management questions/commands in natural language via a voice interface. It uses the Alexa intent service to help map user input to the designated back-end method to perform the appropriate low-level actions on the behalf of the developer. Developers can now offload repetitive Git/GitHub related tasks to Robin, all without any context required. Robin currently supports a number of issues, pull requests, and source code management questions/commands, which we will go over in Chapter 4.

Robin can help serve as an interface between the developers and the GitHub service to help with repetitive or predefined day-to-day tasks. In this work, we investigated when and how a software developer would interact with such tools. To evaluate Robin's usability and potential use cases in software development settings, we conducted a small study consisting of 10 participants. These participants were given a number of scenarios where it was appropriate to ask Robin for help. Each person then executed multiple tasks with Robin to get answers for each scenario. The study showed that our participants were able to successfully interact and ask Robin for answers to specific questions with respect to GitHub. Still, there were a lot of downfalls and potential problems that come with a voice assistant trying to perform such important, software-related tasks. For example, speech recognition and speech-to-text transcription are still on-going problems that require much more research and technological advancements. However, the idea of Robin still showed great potential to be adopted in the software development environment.

Throughout this thesis, we will be going over the related works and background of Robin in Chapters 2 and 3, implementation in Chapter 4, and different use case scenarios in Chapter 5. Our user study and their respective results follow in Chapter 6. Finally, we summarize our findings and potential future work in Chapter 7.

Chapter 2

RELATED WORK

As our project idea of developing a voice-assistant to help developers manage their code repositories is very specific and unique, there were not many related works that aligned with our idea completely. Despite being a very specific idea, there are still some related works that pertain to voice assistants and automated helpers for software developers in general. We will be discussing several of them.

As mentioned in [7], there were a couple attempts at creating voice-assistants for developers. Whether that be for educational purposes or actual software engineering automated help, online marketplaces such as Amazon and Google carry all sorts of applications.

We began our research by investigating Siri Shortcuts, Google Actions on the Play Store, and Alexa Skills on Amazon to see if our idea has been implemented and published in these popular voice-assistant marketplaces. Siri Shortcuts did not have anything like Robin. Most shortcuts just combine a series of actions for commands like creating a schedule on the calendar, sending an email, and adding a reminder. Custom shortcuts similar to Robin, would have to be created by GitHub specifically. If GitHub were to add shortcuts for their app to commit, push, or create issues/pull requests, then that would be just like Robin. The Play Store and the Amazon Skills Store also rarely have any actions related to Git/GitHub. The few actions that they had mainly consisted of teaching/reminding the users of how to use Git commands like committing, pushing, and pulling. However, there was one Alexa Skill that came closest to our idea of Robin. *Lab Assistant*[1] works with GitLab repositories to support questions related to issue-tracking, task-management, and source code management. Our project follows a similar flow of conversation and questions/commands covered.

Additionally, there were a few peer-reviewed research papers somewhat related to our project idea. In the paper "Towards Providing on-demand Expert Support for Software Developers" [6], da Silva et al.'s project aimed to assist developers in specific situations that they face when solving an individual programming task. An example question for this assistant would be something along the lines of "how to use the syntax for the script HTML tag". However, their supported queries didn't cover the software project collaboration realm like what our project is doing. In addition, because this project is purely a chat-based assistant, it did not have to go through the unstable voice-to-text process. Even though [6] did not provide similar support for software developers, we still learned from their implementation steps and information gathering to collect what tasks developers generally needed the most help with.

In another paper called "Msrbot: Using bots to Answer Questions from Software Repositories" [3], the authors proposed a text-based bot to support software developers in answering questions related to software repositories. This project was implemented with the help of Google's Dialogflow. Although their work didn't attempt to create a voice interface, it shares similarities with our approach to creating a source code management assistant. The set of questions they support are related to collaborative software repositories that developers engage in. Those tasks include bug fixing, issue tracking, and working with multiple commits from different contributors on a project. They also provided a list of 15 supported questions deemed relevant based on the software engineering research literature, which is very useful for our selection of questions to cover as well. Furthermore, to test their work, they also included a

[1]https://www.amazon.com/Nathan-Friend-Consulting-LLC-Assistant/dp/B07XY2NBQC

case study with 12 participants. In Chapter 6, we also carried out a small study with 10 participants to determine the usability and value of our idea in real world settings.

The paper "Context-aware Conversationally Developer Assistants" implemented Devy [5], a voice controlled application, to help developers offload low-level actions to an automated assistant. This assistant was implemented on the Amazon Alexa platform. Devy's approach supports context-aware actions, assuming the developer is sitting at their machines with a task associated with a particular context on their workstation. This application is similar to our project idea; however, it doesn't align with our intention to help offload the developer from repetitive tasks. Users shouldn't have to get into the "context" on their workstations to begin with. However, this paper's goals and research interests still highly align with ours. Their research questions help set the foundation and purpose for Robin. In addition, Devy was evaluated on 21 experienced industrial developers. Our small study followed a similar study set up, as will be shown in Chapter 6.

Unlike [6] or [3], our work aims to provide a voice controlled application rather than a text-bot or chat-bot. We decided to not focus our commands on bug-tracking like in [3], but rather, we paid more attention to issue/task tracking and added the ability to manage the desired software repository similar to Devy in [5]. Robin supports questions that can be used by developers and teams while collaborating synchronously or asynchronously without looking at context data loaded on the developers' machine. In contrast to Devy, Robin does not require developers to be sitting at their computers, providing contextual data based on the files they have open on their IDEs or repositories and branches they have checked out. Whether it be during team meetings or independent work times, developers can rely on Robin to manage repositories, branches, and pull requests as well as track issues and tasks for the repository of interest without any context.

Chapter 3

BACKGROUND

The use of voice-assistants have grown tremendously in popularity over the past couple years. With the top tech companies all contributing to the advancements of the artificially intelligence field, personal assistants are now available to help alleviate our small, repetitive daily tasks. Voice assistants have the ability to help us turn off lights, remind us that our best friend's birthday is coming up, and even entertain kids by answering their silly questions.

However, these powerful hands-free tools are nowhere near perfection. Speech recognition and speech-to-text transcription are extremely difficult tasks to be accomplished. There are a lot of factors that go into a machine's sound wave recognition. Background noises, echos, accents, similar sounds/words, machine error, and disorganised speech are all contributing factors.

Chatbots, on the other hand, help strip all these difficulties away. However, this also means that we are passing on the hands-free design of voice-assistants. In this thesis, we decided to keep the hands-free architecture and proceeded with the voice interface for our assistant. For the development of a voice assistant, there are two main platforms, Google and Amazon, that we could implement our application on.

3.1 Google Assistant vs Alexa

Robin was initially implemented on both the Google Assistant and the Amazon Alexa platforms. The two platforms provide very similar architectural settings for developers to create a personalized assistant for different purposes. The only difference is the conversational user interface for the natural language understanding (NLU) platform. The Google Assistant “Action” requires the help of Google’s Dialogflow, whereas the Alexa “Skill” is built through the Alexa Developer Console platform. Some terminologies also differed, but both platforms ultimately produce very similar assistants as shown in Figure 3.1. First, the user states a question/command to the assistant. The NLU platform then tries to match that question/command to an intent and sends a webhook request to our custom Skill/Action’s back-end. The back-end could interact with other external APIs or databases to generate a response. That response is then sent back to the NLU platform to be processed and output to the end user via voice.

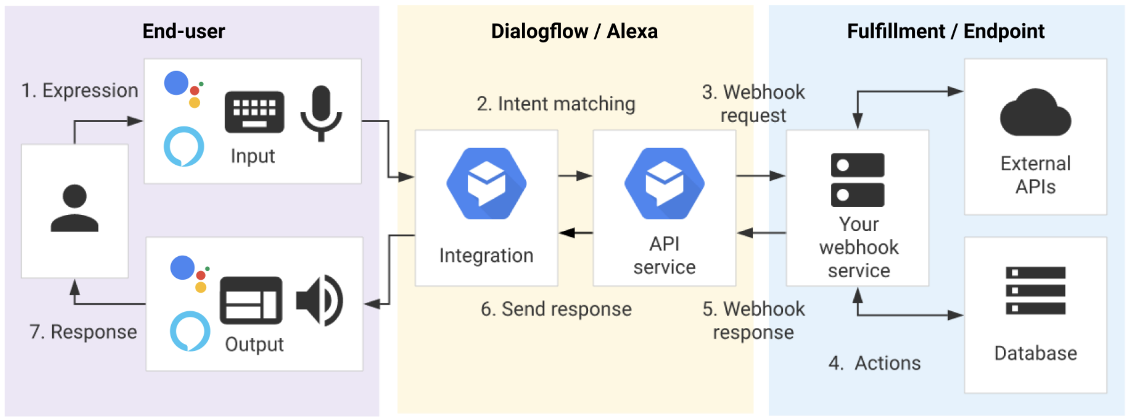


Figure 3.1: High-Level Architecture of Alexa Skill and Google Action

The two personalized assistants overall provide very similar services for our GitHub-based commands. However, in continuation from the previous iterations of the both Google’s and Alexa’s GitHub voice assistants, we decided to evolve the Amazon Alexa version of Robin since the code base for that version was better structured.

3.2 GitHub-based Commands

The purpose of this paper is centered around a voice-assistant that can quickly and effectively provide answers to or perform actions on the behalf of individual developers and teams. Robin has the ability to interact with source code and tasks management tools for developers' software projects. There are many source code and tasks management platforms, but the key platform we decided to focus on in this research is GitHub. GitHub provides both the API and the web interface for developers to get things done. With 72% of the Fortune 50 companies using GitHub Enterprise and over 56 million total developers contributing on GitHub from all around the world [2], it is one of the most popular tools in this field of software development today.

3.3 Terminology

As mentioned earlier, the Google Assistant and Alexa can provide similar assistants. However, they each use different platforms, NLU/NLP models, and of course, different terminologies. We will focus on the terminologies that relate to building this specialized assistant for software developers on the Alexa Developer Console. In the Table 3.1, different vocabularies are defined with concrete examples for reference throughout this paper for a better understanding of how Robin was designed and implemented.

Robin is an Alexa *Skill* that assists developers with GitHub interactions regarding issue-tracking and software maintenance. The user can speak an *utterance* such as "last build's status" to trigger the *GetBuildStatusIntent* intent for a response. While figuring out the response, important variables and their values are kept as *session* or *persistent attributes*. Session attributes are temporary like a pull request number,

Table 3.1: **Alexa Development Terminologies**

Term	**Definition**	**Examples**
Skill	Applications that allow more usage with Alexa	Robin Teammate
Intent	An action that fulfills a user's spoken request	GetBuildStatusIntent
Utterances	Set of likely spoken phrases mapped to the intents	"build status of pr <pr_num>" "last build's status"
Intent Slots	Important variables from the utterance	pr_num
Session Attributes	A key-value data structure to store information on subsequent requests in the Skill's session	{ pr_num: 24 }
Persistent Attributes	Similar to Session Attributes but these data persist beyond the lifecycle of the current session	{ user: "facebook", repo: "react" }

whereas persistent attributes are stored in a database for future sessions like repository information such as "facebook"/"react".

Chapter 4

IMPLEMENTATION

As we described in the previous chapters, this work evolved the Robin voice-assistant in the Amazon Alexa platform. In this chapter, we focus on describing how we have designed and implemented Robin. As an overview, Robin was developed on Node.js via JavaScript with the help of Alexa Skills Kit (ASK) SDK. Robin's HTTP webhook response's endpoint service is currently being hosted on Heroku [1].

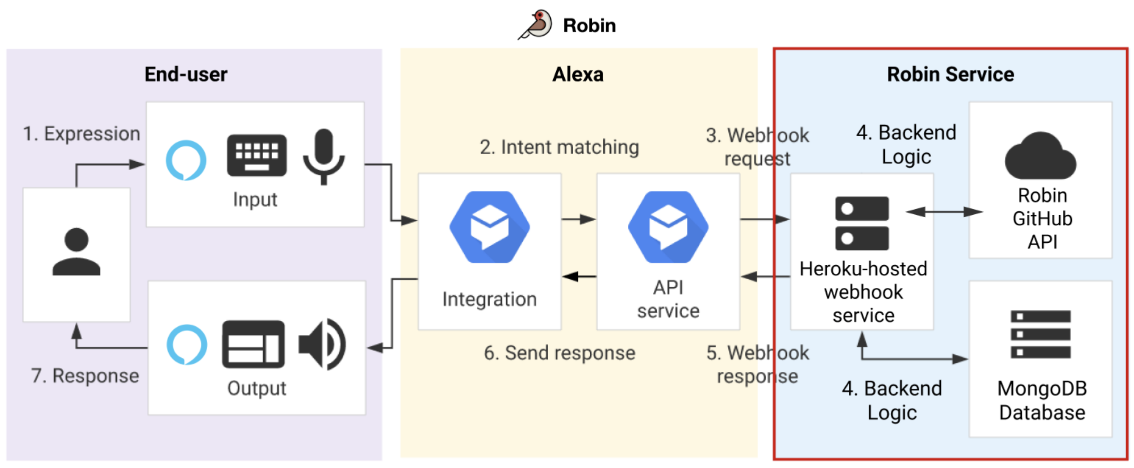


Figure 4.1: Architecture of Robin

4.1 Architecture

Our architecture is similar to the one mentioned in [7] as well as our Alexa vs Google comparison in Figure 3.1. Our main contribution of Robin has been done in the red-outlined box, the *Robin Service* section of Figure 4.1. There are three main components to Robin. The *End-user* component is where the user speaks to and

[1]https://voice-for-devs-robin.herokuapp.com/

receives a response from the Alexa component. The *Alexa* component is where the user's input is parsed with the NLU/NLP models and mapped to the appropriate intent. The third component, the *Robin Service* component, is where our main back-end processes lie.

4.1.1 Alexa Component

Although Alexa's machine learning model, natural language processing, and natural language understanding pieces are out of our hands, we can still control intent matching. The extent to which we can control intent matching is to add a lot of different utterances that can be mapped to their designated intent. We were responsible for manually adding all the intents with their numerous utterances to the Alexa component to train a good model. The supported intents are all shown in Table 4.1, a couple pages later.

The Alexa component then sends a request to our back-end service once it is able to successfully map a user's input to a specific intent. The request is sent as a POST request where the JSON body contains all the parameters necessary for the service to perform its logic. Our Robin service then handles all the calculation/actions and generates a JSON-formatted response related to our supported issue-related and source code management tasks. Once completed, the response is passed back to the Alexa component to be spoken to the user.

4.1.2 Robin Service Component

Our Robin service is hosted on Heroku and our back-end functions are performed with the help of dependencies such as the Robin GitHub API and the MongoDB database when called upon.

4.1.2.1 Back-end Service Design

Many designs and implementation decisions were made throughout the implementation process. Below are a few important design choices that we made to help with Robin's communication flow and usability.

- **Repository Selection**: The repository selection happens at the start of the skill. Once the user successfully enters the Robin Skill, Robin will inform the user about the repository selection. The repository is either already selected (cached from last interaction with Robin), or the user has the choice to select a new repository. When selecting a repository, Robin will prompt the user to first say the name of the repository the user wants to interact with, followed by the name of the repository owner. For example, with the GitHub repository on https://github.com/facebook/react, *facebook* is the owner and *react* is the repository.
- **Speech to Text**: Due to the fact that GitHub usernames are not necessarily regular English words, it is difficult for the Alexa speech recognition model to understand seemingly random words. Therefore, we decided to allow our users to spell out words for parameters such as usernames and issue labels. When taking in repository and owner names while selecting a repository to interact with, Robin will mainly require the user to spell out all names. In addition, we also had to parse all the spelled out punctuation words to their specific punctuation as well. For example, when the user says *underscore* or *dot*, we convert it to the actual punctuation characters "_" and "." respectively.
- **Access Control**: In order to make changes to a GitHub repository via the GitHub API, an authorization token with access must be passed with the HTTP methods. We initially proceeded with the OAuth to allow different users to make

changes to the GitHub repository through their personal accounts. However, we were unable to get it to work well enough and decided to revert to having a general *robin-teammate* account that interacts with the repository instead. We are requiring the user to manually add robin-teammate as a collaborator with push access in order to use the Robin Skill. Although we understand that this is troublesome and potentially insecure, it is just our current work-around.

- **Giving User Selections**: There are different cases where Robin will need more information from the user in order to fully support the user's commands. Rather than forcing the user to repeat the command with all the necessary information off the top of their heads, Robin can list possible choices for the users to select from. For example, the user may want to ask for a GitHub build status, but doesn't know the specific pull request number. Robin will give the user all the open pull requests that are currently in the repository and ask the user to select one of the numbers it had listed out. Another example is when searching for a file in the repository. There may be same file names in different paths of the repository. Therefore, the user is given the different choices to select from to help Robin select the right file to perform the intended actions.

- **Asking for Confirmations**: When dealing with important GitHub actions, we also made sure to ask the user for confirmation before completing the command. For example, when Robin is asked to merge a pull request, it will make sure there is no conflict and then asks the user to confirm the merge. This will give the user a chance to think twice if necessary, and avoid mistakes if the merge is not their intended command. In addition, when Robin gives an answer with a long list of elements as a response, it will only first name a few. Then, it will prompt to check whether the user would like to hear more, if yes, Robin will continue to list the answers, if not, the intent ends.

4.1.2.2 MongoDB Database

The ASK SDK provides an interface for custom implementation to retrieve and save attributes to a persistence layer (i.e. database or local file system). We could have implemented our own "Persistence Adapter", however, there is already an implementation for the MongoDB database. The *ASK-SDK-MongoDB-Persistence-Adapter* package is an add-on package for the core SDK. There are also other Persistence Adapters in the SDK for the AWS services such as the AWS DynamoDB Persistence Adapter and the AWS S3 Persistence Adapter. But because we are not hosting our skill on AWS, we did have access to these cloud storages directly.

Therefore, we decided to move forward with the MongoDB Persistence Adapter. We specifically are hosting our data on the cloud-based, MongoDB Atlas. MongoDB Atlas provides an easy way to host and manage data in the cloud. They also allowed the deployment of a free tier cluster, perfect for our small-scale hosting of data for this thesis.

The MongoDB Database is used in Robin as a means of storing the last repository the user has last interacted with. As described in Table 3.1, persistent attributes are data that persist beyond the life-cycle of the current session. We decided to store the repository and owner names in our own MongoDB database. This is mainly due to the difficulty that comes with having to spell out the repository and owner names. It is often troublesome to set up the repository before getting to fully interact with it. Therefore, we decided to help ease this problem by saving the repository information for the next sessions. However, the user can still opt to select a different repository when they launch the Robin skill.

The data is stored with the primary key userId. The userId string represents a unique identifier for the Amazon account for which the skill is enabled. This will help Robin keep track of the different user account's selected repository. However, it should be noted that if the Amazon account disables the Robin Skill, the userId will no longer be the same and they will lose access to their previously used repository.

4.1.2.3 Robin GitHub API

In order to perform GitHub functions, we must interact with the GitHub API. Previously, we have been calling the GitHub API via HTTP methods separately with the help of the axios [2] library. However, we realized that interactions with GitHub related functions such as issue-tracking and source code management could be extended even further. For example, other platforms such as the Google Assistant, Slack, and Discord could be used to ask questions related to a software repository as well. Thus, with the help of another student who is currently doing individual research related to a Discord GitHub text-bot, we decided to create a wrapper for our issue-related and source code management GitHub HTTP calls and expose that wrapper as a REST API. This became the *Robin GitHub API* component of Figure 4.1, and it is currently hosted on Heroku [3] with the Robin Teammate access token as well. This means that robin-teammate, again must be added as a collaborator with push access in order to perform actions in the specific repository.

4.1.3 Supported Commands

Most of the commands are related to issue-tracking and basic source code management that are often done through the GitHub web interface or via the command-line. The

[2]https://www.npmjs.com/package/axios/

[3]https://robinrestapi.herokuapp.com/api-docs/#/

supported commands are listed in Table 4.1. Through Robin, developers are able to accomplish these tasks with just their voice.

4.2 Robin Deployment

As mentioned earlier in the chapter, Robin's back-end application is currently deployed on Heroku. The skill can now communicate with this webhook service at any time. However, when it comes to our skill, it is currently only available through accessible accounts that we have added. There is also an option for beta testing with a limited group of customers as a quality assurance test. In order to submit and publish our skill to the Amazon Skills Store, we must review Amazon's submission checklist.

Amazon requires the skill to pass a certification process first before publishing it to customers on the Alexa Skills Store. There is a long checklist for certification of a custom skill. This includes following the Alexa policy guidelines, security requirements, and performing all functional/voice interface and user experience tests required. The Alexa Development Console provides support in validating whether the skill is on the right path. Once the skill is certified, Alexa will then proceed with publishing the skill to the Alexa Skills Store. We have the option to publish right after certification, or manual select to do so afterwards. Robin is currently in the certification process as this thesis is written.

Table 4.1: Robin GitHub Commands

Intent Names	**Description**	**Input Example**
GiveRepoName	Sets the user's GitHub repository that Robin will interact with	"select <repo name>"
ListRepos	List all the other repositories in the repo the user chose to interact with	"list all repos"
GetLabels	Lists all the available labels for the issues in the repository	"list all labels"
GetIssues-WithLabel	Lists all the open issues with a specific label	"issues with label <label>"
GetIssues-Assignees	Lists all the developers assigned to a specific issue	"assignees of issue <num>"
NumAssigned-OpenIssues	Gives the number of open issues in the repository	"open issues that are assigned"
GetAssignee-Issues	Lists all the issues assigned to a specific developer	"list all issues assigned to <user>"
GetNumPRs	Gets the number of open PRs in the repository	"how many open PRs are there"
GetPROwners	Lists the owners of all the PRs that are currently open in the repository	"people with pull requests open"
GetMedian-ReviewTime	Gets the median review time of all PRs within a certain time range	"what is the median PR review time over the past three weeks"
MergeBranch	Merges a specified head branch into specified base branch	"merge <head>to <base>"
MergePR	Completes the pull request and merge as necessary	"merge PR <num>with message <msg>"
CreateIssue	Creates a new issue in the repository	"create an issue"

Intent Names	**Description**	**Input Example**
AddUserToIssue	Adds a user to an issue (based on issue number)	“assign a user to issue <num>”
AddLabelToIssue	Adds a user to an issue label (based on issue number)	“add a label to issue <num>”
CloseIssue	Closes a specified issue (based on issue number)	“close issue <num>”
CreateIssue-Comment	Adds a comment to a specified issue (based on issue number)	“comment on issue <num>”
GetReviewers	Gets all the developers who has reviewed a specific PR	“who reviewed pull request <num>”
GetOldestIssue	Gets the oldest open issue in the repository	“what is the oldest issue”
CreatePR	Creates a PR from specified head branch	“create pr for <head>”
ApprovePR	Approves a PR (based on PR number)	“approve PR <num>”
GetLast-Contributor	Gets the last contributor to a specified file	“who was the last to edit <file>”
GetUnassigned-Tasks	Lists all tasks that are open and unassigned in the repository	“what unassigned tasks are in the backlog”
GetBuildStatus	Gets the build status of a specific PR	“what’s the build status of PR <num>”

Chapter 5

USE CASE SCENARIOS

Robin was developed with both individual developer and software teams usage in mind. Whether it be at a team meeting or an individual developer's workstation, Robin could assist software engineers with their repetitive issue-related tasks as well as source code management tasks. There are a variety of situations and scenarios where this voice-controlled assistant could be of great use to a developer. Different commands map to different intents for different purposes depending on the context and the user's specific needs. In this chapter, we will be going over some of the possible scenarios where Robin could benefit a developer. These use case scenarios are based on the supported commands represented in the previous chapter and in Table 4.1.

5.1 Individual Developer Use Cases

For the different use cases in this section, we will focus on Developer A.

1. **Checking Build Status of Pull Request**:

   Scenario:

   Developer A is in the middle of a research task with various tabs open. It is very difficult to navigate away from all those research tabs of information. However, he/she has to check if the build of his/her recent pull request has passed. This would mean that he/she would have to switch context from research to GitHub.

   Robin Command:

Developer A can ask Robin to find out the build status of his/her pull request with the GetBuildStatusIntent.

Example Input:

"Alexa, launch Robin teammate and tell me the build status of PR #24".

2. **Checking Last Contributor of a File**:

   Scenario:

   Developer A is currently very focused on his/her implementation on an IDE. While looking through File Z, he/she discovered that there is a potential bug in this specific file. He/she needs to pinpoint who was the last to make edits to this file to determine the next steps. However, this would require going into the command line or the GitHub web interface to find out.

   Robin Command:

   Developer A can ask Robin to find who was the last to make edits to File Z with the GetLastContributorIntent.

   Example Input:

   "Alexa,launch Robin teammate and tell me who was the last to make changes to the test.txt file".

3. **Checking Issues Assigned to Developer A**:

   Scenario:

   Developer A is in the middle of a research task with several other tabs open. It is very difficult to navigate away from all those research tasks, but he/she has to check what other tasks he/she has to do for the current sprint other than the current research task to help time manage efficiently. Again, this would require Developer A to switch context from research to GitHub.

   Robin Command:

   Developer A can ask Robin to find which issues are assigned to him/her with the GetAssigneeIssuesIntent.

Example Input:

"Alexa, launch Robin teammate and list all issues assigned to A".

4. **Merging of a Pull Request**:

   Scenario:

   Developer A is currently contributing in a discussion on the team's online thread. He/she suddenly received a notification from colleagues that the pull request he/she has sent out is approved by every reviewer requested. Developer A will have to drop his/her current context on the discussion thread and merge the pull request. However, this would mean that he/she has to switch context from forums such as Slack or Microsoft Teams, to GitHub.

   Robin Command:

   Developer A can ask Robin to merge the pull request on he/her behalf with the MergePRIntent.

   Example Input:

   "Alexa, launch Robin teammate and merge PR #24".

5. **Creating an Issue**:

   Scenario:

   Developer A was implementing deep inside the software repository for this sprint. As he/she finishes one small feature, he/she realizes that test cases still need to be added. However, Developer A was sent a meeting invite that is about to start. He/she would have to switch context from the code base to GitHub to add an issue to remind him/her.

   Robin Command:

   Developer A can quickly ask Robin to create an issue to remind him/her about the test task with the CreateIssueIntent.

Example Input:

"Alexa, launch Robin teammate and create a new issue".

6. **Creating a Pull Request**:

   Scenario:

   Developer A had just finished pushing their changes to GitHub via command line. He/she pushed the changes to his/her working branch named "feature". Before moving on to adding new code changes, he/she wants to create a pull request to get their changes reviewed and integrated to the main branch. However, this would require him/her to switch context from the code to the GitHub web interface.

   Robin Command:

   Developer A can quickly ask Robin to create a pull request from "feature" to "master".

   Example Input:

   "Alexa, launch Robin teammate and create a pull request".

7. **Checking Issues with Label X**:

   Scenario:

   Developer A is currently having a conversation with his manager in a one-on-one meeting. The manager asked the Developer A about their opinion on whether he/she thinks this sprint has too much work compared to the manpower they have. This would require either Developer A or the manager to have to go to GitHub and check the issues for the sprint. However, in a meeting context, it might not be convenient.

   Robin Command:

   Developer A or the manager can ask Robin to find issues with the current sprint label with the GetIssuesWithLabelIntent to continue their discussion.

Example Input:

"Alexa, launch Robin teammate and tell me the issues with the label *sprint*".

8. **Adding Issue Comment**:

   Scenario:

   Developer A is working on a research task. While researching, he/she found an important information about the research information that needs to be recorded on the issue. However, he/she is immersed in a sea of research tabs. To add a comment to an issue, this would require Developer A to switch context from the research tabs to GitHub, which might be troublesome.

   Robin Command:

   Developer A can ask Robin to find and add a comment to the issue of interest with the CreateIssueCommentIntent.

   Example Input:

   "Alexa, launch Robin teammate and add a comment to issue #24".

9. **Approving a Pull Request**:

   Scenario:

   Developer A has just finished reading through the pull request he/she has been assigned as a reviewer. Right away, he/she has been requested to attend a meeting that is about to start. Developer A then left to attend the meeting. After coming back, the developer wants to approve the pull request. But instead of having to find the pull request again, Robin can help Developer A out.

   Robin Command:

   Developer A can ask Robin to help approve the pull request he/she had finished reviewing with the ApprovePRIntent.

   Example Input:

   "Alexa, launch Robin teammate and approve PR #24".

5.2 Software Teams Use Cases

For the different use cases in this Section, we will focus on Team A's sprint planning, stand-up, sprint retrospective meetings. Some of these scenarios are used in our user study in the following chapter as well.

1. **Assigning a Developer to an Issue**:

   Scenario: [Sprint Planning Meeting]

   The team is currently discussing the issues/tasks for the upcoming sprint. They are currently assigning issues for the upcoming sprint. Rather than scrolling through all the existing issues, the team can ask Robin to get specific issues and assign a developer quickly.

   Robin Command:

   The team can ask Robin to add a developer to an issue with the AddUserToIssueIntent.

   Example Input:

   "Alexa, launch Robin teammate and add an assignee to issue #24".

2. **Adding Issue Comment**:

   Scenario: [Sprint Planning Meeting]

   The team is currently discussing the issues/tasks for the upcoming sprint. They had just finished reviewing one task and are now onto the second task of the sprint. However, Team Member A had just remembered that he needed to add extra details on the task that he created. While the team members are deciding the task estimate to vote, Robin can help add a comment for Team Member A's task.

   Robin Command:

   Team Member A can ask Robin to add a comment with the CreateIssueCommentIntent.

Example Input:

"Alexa, launch Robin teammate and comment on issue #24".

3. **Checking Last Contributor of a File**:

   Scenario: [Stand-Up Meeting]

   The Team is going around for updates during a stand-up meeting. During an update, Team Member A discusses a potential bug that he/she found from the task he/she is currently working on. Team Member A then points out that File Z might be the source of the issue/bug. In the a meeting context, it is difficult to find out who last made changes to File Z.

   Robin Command:

   Team Member A can ask Robin to find who was the last to make changes to File Z with the GetLastContributorIntent.

   Example Input:

   "Alexa, launch Robin teammate and find out the last contributor of the test.txt file".

4. **Creating an Issue**:

   Scenario: [Stand-Up Meeting]

   Continuing from the "Software Teams Use Cases" #3. After the last contributor (Team Member B) of File Z was found, Team Member A would like to create an issue to remind that last contributor to investigate the source of the bug.

   Robin Command:

   Team Member A can ask Robin to create an issue to remind Team Member B to look into File Z with the CreateIssueIntent.

Example Input:

"Alexa, launch Robin teammate and create an issue".

5. **Checking Build Status of a Pull Request**:

   Scenario: [Stand-Up Meeting]

   The Team is going around for updates during a stand-up meeting. During an update, Team Member A mentioned that he had just finished reviewing and approved Team Member B's pull request. This means that Team Member B's pull request should be ready to merge. Before merging, we want to make sure that the build status of the pull request successfully passed. Currently in a meeting, it is difficult to take away other people's time to check and merge the pull request.

   Robin Command:

   Team Member A can ask Robin, on the side of the meeting, to check the build status of the pull request with the GetBuildStatusIntent.

   Example Input:

   "Alexa, launch Robin teammate and find out what the build status of PR #24 is".

6. **Checking the Reviewers of a Pull Request**:

   Scenario: [Stand-Up Meeting]

   Continuing from the "Software Teams Use Cases" #5. After finding out that the CI build of Team Member B's pull request successfully passed, Team Member B wants to make sure that his/her reviewers all approve of the pull request. This would mean that the pull request is ready to be merged.

   Robin Command:

   Team Member B can ask Robin, on the side of the meeting, to find out who has reviewed and approved the pull request with the GetReviewersIntent.

Example Input:

"Alexa, launch Robin teammate and check who has reviewed PR #24".

7. **Merging a Pull Request**:

   Scenario: [Stand-Up Meeting]

   Continuing from the "Software Teams Use Cases" #5 and #6. After finding out all the reviewers approved Team Member B's pull request, it is now ready to be merged. Instead of waiting until after the meeting and risk getting distracted with other events, Team Member B can just merge the pull request right away.

   Robin Command:

   Team Member B can ask Robin to help merge his/her pull request with the MergePRIntent.

   Example Input:

   "Alexa, launch Robin teammate and merge the PR #24".

8. **Close/Create an Issue**:

   Scenario: [Stand-Up Meeting]

   Continuing from the "Software Teams Use Cases" #5, #6, and #7. Once the Team Member B's pull request is merged, he/she can close the issue associated with that task. If there is a merge conflict, then create another issue to fix the merge conflict.

   Robin Command:

   Team Member B can ask Robin to either close or create an issue with the CloseIssueIntent or CreateIssueIntent.

   Example Input:

   "Alexa, launch Robin teammate and close issue #24".

9. **Checking for Unassigned Issues**:

   Scenario: [Stand-Up Meeting]

The Team is nearing the end of the sprint. After everyone's updates during the stand-up meeting, Team Member A mentioned that they have extra bandwidth for the sprint as they had already finished all their asssigned tasks. Instead of having someone navigate through the backlog to find unassigned issues, Robin can help with that. This will help everyone in the team be more aware of possible extra tasks if they have extra bandwidth for the sprint as well.

Robin Command:

Team Member A can ask Robin what unassigned tasks are in the backlog with the GetUnassignedTasksIntent.

Example Input:

"Alexa, launch Robin teammate and tell me what issues are still unassigned".

10. **Assigning a Developer to an Issue**:

Scenario: [Stand-Up Meeting]

Continuing from the "Software Teams Use Cases" #9. Once Team Member A finds all the unassigned tasks in the backlog. He/she can then assign him/herself to the issue. Again, instead of having someone navigate through GitHub, Robin can quickly help with the issue assignment. Also, doing the assignment in the meeting helps everyone be aware of what Team Member A is up to for the remainder of the sprint since this is new information (after Team Member A's stand-up update).

Robin Command:

Team Member A can ask Robin to assign the unassigned issue to him/herself with the AddUserToIssueIntent.

Example Input:

"Alexa, launch Robin teammate and assign A to issue #24".

11. **Checking the Median Review Time**:

    Scenario: [Sprint Retrospective Meeting]

    The team is currently in a sprint retrospective meeting, discussing about the quality of their latest sprint. One measure that would be excellent for the team to know about their velocity is the amount of time it takes to review pull requests. The team would rather be developing more features than reviewing other people's pull requests. If it takes a long time to review a pull request, additional attention should be given to this fact. It might be the case of low quality PRs requiring longer times to get approved, or too big PRs needing to be broken down, or limited resources to dedicate to code review, etc. Therefore it is important to have this metric during sprint reviews, but it isn't usually part of the sprint metrics. Robin can help answer the median time it takes to review pull requests.

    Robin Command:

    The team can ask Robin what the median review time is over the span of the sprint with the GetMedianReviewTimeIntent.

    Example Input:

    "Alexa, launch Robin teammate and tell me the median PR review time since two months ago".

12. **Finding the Oldest Issue**:

    Scenario: [Sprint Retrospective Meeting]

    The team is currently in a sprint retrospective, discussing their backlog from the sprint and which tasks to carry over to the next sprint. While doing so, the team notices a lot of older tasks that have not been addressed for a while. Instead of searching of the oldest issues that are of technical debt, the team can quickly ask Robin to find that.

    Robin Command:

The team can ask Robin to find the oldest issues that might be building up technical debt with the GetOldestIssueIntent.

Example Input:

"Alexa, launch Robin teammate and tell me the oldest issue".

In this chapter, we presented a couple use cases of Robin. These uses cases from the "Software Teams Use Cases" section was used as part of the user study we conducted. In the next chapter, we will share this user study and share its results.

Chapter 6

USER STUDY

To study the usability of Robin, we conducted a small study with 10 participants. We initially wanted to do an official experiment with A/B testing. However, due to the time-constraints of this project and the current work from home situation, we decided to conduct a less formal and less controlled user study, which we will describe in this chapter.

6.1 Research Questions

In the evaluation of our user study, we will be examining the follow research questions:

RQ1 What are the positive and negative aspects we can observe from using a voice-assistant like Robin?

RQ2 To what extent can a voice-assistant support basic software development tasks related to issue-tracking and source code control?

RQ3 To what extent is Robin feasible in supporting software developers?

6.2 Study Design

1. **Gather Participants**

   Due to the COVID-19 circumstances at the time, we proceeded with convenience sampling as a way of recruiting our participants rather than random sampling. Participants were people that we knew personally and were selected based on our knowledge of the people and their software engineering experiences. Those software experiences required from the participants are as follows:

- Participants should have experience in the software industry
- Participants should be somewhat familiar with GitHub and git commands

2. **Participants Study**

   Part 1: Schedule and meet with each participant individually over Zoom. Describe the project overview and explain to the participants what Robin is implemented for. Describe the need to spell out usernames. Make sure the user understands that the Robin skill might need to be launched again as timeouts may occur.

   Part 2: Guide the participants through each study question. The study was conducted one-on-one as we observed the Robin interaction throughout the entire meeting. We describe three main scenario cases and ask each participant to ask Robin for help as necessary. More details on each scenario are provided in the Section 6.2.2 below.

   Part 3: Ask the participants the exact same questions like in Part 2. However, instead of asking Robin to provide answers or perform actions on the participants' behalf, the participants will now perform those tasks manually on the GitHub web interface. We asked the same participants to complete Part 2 and Part 3 mainly because of the limited number of participants given the virtual situation. In addition, using the same participants for the voice-controlled versus manual GitHub scenarios enabled our participants to freely compare their experiences on both ends.

3. **Analysis**

   Number of Attempts: We attempt to analyze the usability with quantitative data by counting the number of attempts it took a participant to get the right answer from Robin.

Post-Study Survey: As for our qualitative data, we sent out an anonymous, post-study survey to our participants to gather their honest opinion about Robin after having used it in comparison with the manual, web-interface interaction. More details on the survey is provided in Section 6.2.3.

We conducted this study with the iOS/Android Alexa Application on the participant's mobile device over a Zoom call. Everyone is set to interact with our test repository[1]. All participants are also given the same Alexa login account. This is mainly to interact with the test repository we set up for the study. In addition, by having a test login account, we relieve the participants of having to create their own accounts, which might be bothersome since we already have to ask them to install the Alexa Application.

6.2.1 Participants

We did convenience sampling of participants for our project user study. We mainly asked people that we know and have software development background. We also made sure they are familiar with what GitHub is and have used GitHub before through some screening questions.

From the participants that we were able to recruit, we got an even split of 5 males and 5 females. They were all either fourth year and graduate computer science/software engineering students or already working in the industry. Three of the participants are or will become project managers after graduation.

[1]https://github.com/mmachiya/feather/

6.2.2 Robin Interaction Questions

Because we wanted to focus on the usability of Robin in a software team setting, we decided to ask the participants to interact with Robin while pretending to be in a software team stand-up meeting. Please note that during a team stand-up, developers are often all away from their machines. This makes it hard to make immediate issue or source code management related changes.

These questions are the same as the Software Teams Use Cases #2 through #9 in Chapter 5.

1. **Scenario 1**

   During an update, a team member discusses a potential bug and points out a file *recommendation.swift* that might be the source of the issue.

   Participant Action: [GetLastContributorIntent]

   Find out who was the last person to make changes on recommendation.swift file. (Software Teams Use Cases #2)

   Participant Action: [CreateIssueIntent]

   After finding out the last contributor, create an issue to assign that developer to take action after the meeting. (Software Teams Use Cases #3)

2. **Scenario 2**

   Moving onto a different teammate's update. This team member mentioned that they finished reviewing and approved another teammate's pull request [pr_number]. This pull request is related to issue [issue_number].

   Participant Action: [GetBuildStatusIntent]

   Check the build status of pull request [pr_number] to make sure the pull request passed the CI build. (Software Teams Use Cases #4)

   Participant Action: [GetReviewersIntent]

After finding out that the build passed, check the reviews of pull request [pr_number] to ensure that it is ready to merge. (Software Teams Use Cases #5)

Participant Action: [MergePRIntent]

After finding out who reviewed the pull request, go ahead and merge to pull request [pr_number]. (Software Teams Use Cases #6)

Participant Action: [CloseIssueIntent / CreateIssueIntent]

If the merge was successful, close the issue [issue_number] related to the pull request. If the merge has conflicts, create a new issue to fix the conflicts. (Software Teams Use Cases #7)

3. **Scenario 3**

   After finishing everyone's update, the team is now in post-scrum. One team member, [username], mentions that he/she has extra bandwidth to do more work during the sprint.

   Participant Action: [GetUnassignedTasksIntent]

   Find all the possible unassigned tasks left in the sprint. (Software Teams Use Cases #8)

   Participant Action: [AddUserToIssueIntent]

   After getting a list of unassigned tasks, assign [username] to one of those issues listed. (Software Teams Use Cases #9)

6.2.3 Robin Study Survey

After having the chance to interact with Robin through our Study, we want the participants' opinions on their experience. We conducted a Google Forms survey and sent it out to our participants after their individual studies. There are two main sections to our survey: previous GitHub Experience and Robin Experience.

1. Previous Github Experience

    - How familiar are you with GitHub?
    - Have you used GitHub for your work in the software industry? Y/N
    - Have you used GitHub for school projects? Y/N
    - Have you used GitHub for personal use like personal websites or for repositories of projects? Y/N

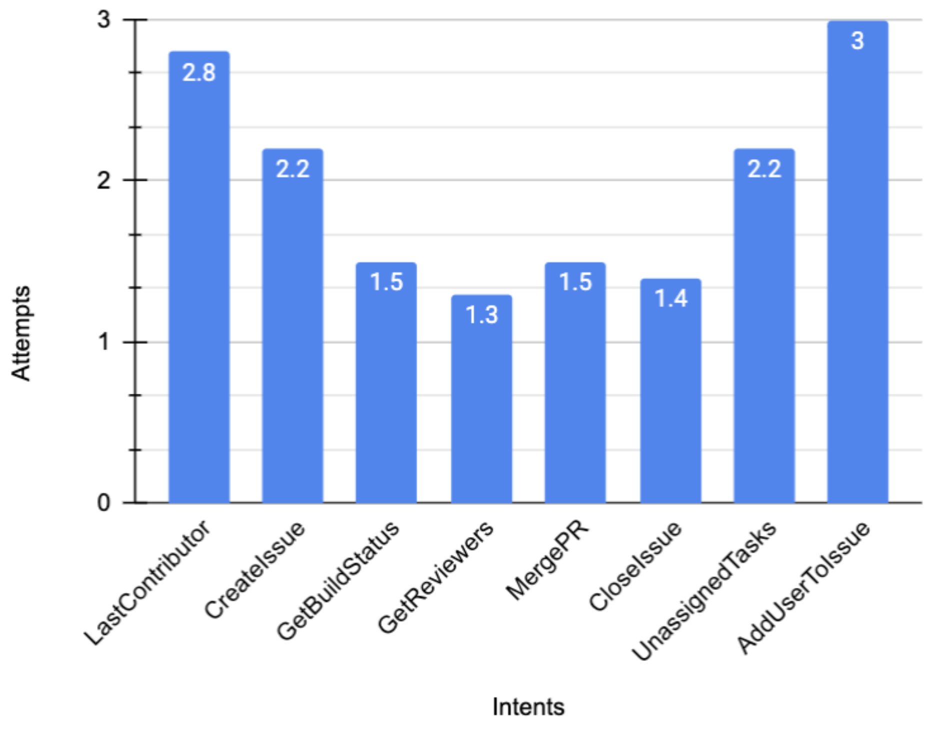
2. Robin Experience

    - How was your experience interacting with Robin? (Positive? Negative? Why?)
    - What part of Robin was hard to use?
    - What part of Robin was easy to use?
    - Are there any other tasks / workflows that you think Robin could help you with? (for different scenarios?)
    - What part of Robin could be improved from your interaction with it?
    - Imagine you're using Robin in your work environment as part of a team, how would that be? Please describe.
    - Why, how, and to what extent do you think Robin would add value to you as a software engineer?

6.3 Study Results

After finishing our one-on-one study with all 10 participants, we evaluated the usability of Robin in two different measures. One being the number of attempts asked to Robin to get the expected answers as mentioned in the Section 6.2.2. Another measure is a qualitative review of the users' individual experiences and opinions after interacting with the voice-assistant as mentioned in Section 6.2.3.

Figure 6.1: Questions/Commands Attempts Chart



We first calculated the number of attempts by averaging the attempts over all our participants attempts for each command. As shown in Figure 6.1, it took the participants quite a few attempts to get the right answers from Robin. On all the commands that were asked during the study, it took over 1 attempt on average. The highest number of attempts being the task to add an assignee to an issue. This is due to the fact that we have to spell out the user names as it is hard for the Alexa ML model to understand uncommon pronouns or different developer aliases via voice. Many participants also forgot to spell out the username at first, so they had to re-attempt the task.

The second highest number of attempts was from the first command we asked the participants to ask Robin, which was getting the last contributor to a specific file. This is mainly because the participants were initially unfamiliar with the flow of Robin at first. In addition, the task required the user to say a specific file name. Many participants stated the file name wrong. For example, we asked the participants to ask about *recommendation.swift*, but a lot of the participants actually said recommendation*s*.swift, which led to an extra attempt on the question asked.

In terms of the qualitative data that we received through the study survey the participants took, we were able to see both the negative and positive side to Robin. From the 10 responses, there was about an even split between having had a positive and negative experience interacting with Robin. Five expressed they had somewhat positive experiences, 4 expressed they had somewhat negative experiences, whereas 1 was a neutral experience. We suspect the reason we had slightly more positive reviews might be because of the participant's personal bias in being acquainted with the experimenter. They might have felt like they needed to say it was a good experience even though the survey was anonymous. Nonetheless, all the participants still gave really good feedback about what parts were hard/easy to use, what could be improved, and whether Robin would be useful in reality. We analyze the positive and negative qualitative aspects in the following sections.

6.3.1 Potential Benefits of Robin

From the survey, many participants commented that Robin did give a good general overview of the state of the project and the team's responsibilities. They liked that they were able to have things read out to them instead of having to scroll through the repository and analyze the necessary information by themselves. One participant even mentioned our purpose for building Robin when they commented that the “hands-free GitHub interaction [feature] could allow for more multitasking and efficiency”.

These survey reviews of Robin made us realize that the idea has the potential to truly benefit developers by being able to carry out small, repetitive tasks like assigning users to issues and closing pull requests. Robin can help people who are busy and whose works are scattered and require different mental models for each. Through simple voice commands, the users can check on something quickly without having to put it on their calendar or to-do list. Many participants also suggested that Robin could be of great use in cases like sprint planning. During sprint planning, it could help with dispersing work quickly and on the fly. Robin would be the one fetching the work and reading it to the team rather than having one person doing that like before.

In addition, Robin can help people who are a little less familiar with GitHub. Some participants who have forgotten the commands on GitHub stated that Robin was useful in that they did not have to refresh their memory on basic GitHub commands. Another great use of Robin would be the case of creating user stories. Since product managers or other similar roles on the team may not be super familiar with the GitHub repository, having Robin help them in that regard could be very useful.

Many ideas for other usages of Robin were also suggested by our participants. We will be going over those in the Future Works section (Section 7.1) later on in this paper.

6.3.2 Limitations to Robin

As shown in the results above, there were some drawbacks related to the use of the Robin voice-assistant. Many of the commands tested during the study needed to be called many times. Not only that, the Robin skill also needed to be re-launched multiple times as well. Although Robin can be launched and asked the question in one command, we did not apply this in the study. Thus, some of the negative comments from the study resulted from the tediousness of having to repeatedly relaunch and say things repeatedly. Another downfall is the nature of voice-controlled assistants not being a visualisable experience. According to [4], only about 30% of the population are auditory learners. This makes it a bit hard for Robin to grow on the rest of the population.

Another negative aspect to Robin is the learning curve. According to one participant, the "learning curve is steep and will likely slow down meetings or development time". This is a fair argument especially since it took the participants quite a few attempts to complete the commands in the study. We think this may be due to the fact that the study was the first time ever for the participants to interact with Robin. Thus they might not be very familiar with the flow of conversation. Every command in Robin follows a similar flow, so after being familiar with that flow, developers should be able to use Robin easily.

One important thing when it comes to voice-assistants is its ability to understand and process the user's statements. An issue that we ran into in this project, is the lack of diverse utterances inputted for the intents. Although the intents tested had at least 20 utterances at time the study was conducted, that was not enough. Some commands that the participants thought were natural weren't trained on the model since we didn't include a similar statement in the utterances. This led to the model not being able to understand the user, and thus more frustration.

All these mentioned hard-to-use aspects of Robin somewhat contradict our intention of having Robin be used for multi-tasking purposes and efficiency. We can attribute the problems with Robin to two main limitations: 1) Alexa's Platform Limitations and 2) Robin Set Up Limitations.

6.3.2.1 Alexa Platform Limitations

This area of limitation is out of our control as we have decided to build our assistant on the Amazon Alexa Platform. The NLU and NLP aspects of the assistant are handled completely by Amazon, and are not open to modifications. Despite our inability to control the ML model and other constructs of the Alexa Platform, it is still important to point out these limitations as they affect our user experience of Robin as a whole.

- **Accents/Pronunciation**: One of the most difficult tasks related to speech-to-text transcription is understanding different accents and pronunciations from people. We do not want to limit our skill's usage to only people without accents, so Alexa's ability to distinguish between various pronunciations of words through different accents is extremely important. Although Alexa does a pretty good job at it, it still had issues with some of our participants with accents. The participants also commented on this negative side of voice-assistants in our survey. One way we tried to counter this problem to have the participant speak a little bit slower, which helped a little bit.

- **Uncommon Pronouns**: As mentioned earlier in the paper, GitHub usernames or developer aliases are not normal words or common pronouns. Therefore, we had to force our participants to spell out the names in certain parts of the interaction with Robin. It was hard for our participants as expected. Many commented that it was hard to memorize the usernames and their spellings. Others also had a hard time when a slightly "long" pause between spelling the different characters cut the users off. We understand that this username aspect is one of the most troubling parts to the voice-assistant. However, if we are able to work-around or find a fix for this uncommon pronoun feature, Robin would be extremely useful.

- **Alexa Timeouts**: Alexa timeouts after 8 seconds if we don't respond to it. One way we handled this is using the re-prompt functionality of Alexa. If Alexa doesn't receive user input the first time, it will re-prompt the user again for the next steps. In addition, if our answer execution time takes longer than 8 seconds, Alexa will think our back-end has errors and automatically send an error response to our user. There are a couple of work-arounds like sending a blank answer or a response saying it is taking a little bit longer than expected. However, the built-in Alexa platform's timeout architecture is something completely out of our control and we had to work with it which limited our flexibility in the conversation flow a bit.

- **Intent Mapping**: The intent mapping process is done completely by the Alexa Platform. It connects users' questions/commands to specific intent functions in our code that handles answer computation or actions related to GitHub. Sometimes certain user input commands are mapped to the wrong intent. Although we are the ones providing utterances as training data for the intent mapping model, we cannot cover all possible variances that a user might say to the assistant. In addition, when dealing with issues and pull requests in GitHub, there are commands that are similar in nature. For example, when trying to apply labels or assign a user to an issue, the Alexa may map to the opposite intent. This actually happened a few times in our study. Distinguishing small differences might be hard for Alexa as people may say things that might be more similar to another intent than the intended one. The intent mapping service depends completely on Alexa's model and architecture for this aspect of the assistant, and is out of our control.

- **Being in the Skill or Not**: It is hard for the users to know whether they are in the Robin skill with the ability to ask about a GitHub repository or not. When asked Robin commands outside of the skill, the responses Alexa gives doesn't provide any information or insight as to the Robin skill. It often confuses the users even more. This is not in our control and is impossible to handle on our end as these events are technically happening outside of our skill and implementation.

6.3.2.2 Robin Set Up Limitations

When we designed Robin and its conversation flow, there are some aspects that are somewhat hard to use. There are a lot more flexibility and options that we could add to help improve how Robin flows and interacts with our users. However, due to a tight timeline, we are unable to make it the most user friendly assistant just yet. Here we will describe some of the limitations that could be improved upon in future works and iterations of Robin that could help improve the user experience.

- **Long Responses**: Some of Robin's current responses are a little bit long as we are trying to give as much information to the user as possible to give them the overview of the state of the repository. However, this is not super user friendly as it forces the user to pay close attention and listen to the responses very carefully for a long time. Some of the comments in the survey mentioned that they had a hard time remembering what Robin said since it was such a long response. They had to listen super closely. This is one limitation that is hard to fix as we are only using voice to give information without any visualization tools.
- **Required Issue/PR Numbers**: One main limitation to Robin right now, which can be fixed with future iterations, is the current requirement of knowing issue/pull request numbers. In order to interact with a specific issue or pull

request, Robin needs a number that is associated with that issue/PR. However, it is hard for users to remember specific numbers and it would be more user friendly if we could map the issue/PR if the user mentions part of the issue/PR title instead. Our current work around is to mention the existing issues/PRs with their titles and numbers for the user to choose from. However, this is a bit inefficient.

- **Better Conversation Flow**: A few comments from our study survey mentioned that the question/answering flow may not be very well “fleshed out”. This is true, as of right now, some intents will ask for specific parameters after asking a question whereas some will take the parameters with the questions asked. We need to be more consistent with our conversation flow and how we take in specific parameters for different intents. The reason for this was the set up of Robin. Because one developer implemented a set of intents whereas another implemented another set without communicating about the user experience and the conversation flow. We were more focused on expanding the number of possible commands rather than the quality of just a few commands. The user study made us realize that, and now we need to focus on making the entire flow and process smoother like how two people would communicate via voice.

6.3.3 Research Questions Discussions

After having implemented and received feedback about Robin, we can now revisit our three research questions.

RQ1 What are the positive and negative aspects we can observe from using a voice-assistant like Robin?

As described in Sections 6.3.1 and 6.3.2 above, Robin has the potential to help increase developer productivity. It can ultimately help automate repetitive tasks that are often cumbersome such as adding labels or assigning a user to an issue. However, there are a lot of future improvements that can be done to fully advance our assistant even further. This could include designing a better conversation flow and allowing specific GitHub users to make changes through Robin rather than using the default *robin-teammate* account.

RQ2 To what extent can a voice-assistant support basic software development tasks related to issue-tracking and source code control?

As of right now, it seems that Robin could be useful in supporting basic tasks like listing out unassigned tasks and adding an assignee/labels to an issue. Many of the comments from our study participants support the idea of using Robin in meetings such as sprint planning to distribute work. However, some participants are still skeptical of the riskier tasks such as merging pull requests to production. Although Robin asks for confirmation, our participants still preferred to do these tasks manually because they get more information about the PR comments/approval on the web page.

In the survey, we asked the participants to suggest specific workflows that would be helpful to developers. The most mentioned tasks are related to getting lists of tasks so the user would not have to scroll through GitHub, or adding assignees/labels on issues for sprint planning. These were the only ones that were mentioned from the study commands specifically.

Other workflows our participants suggested as support for software development tasks implied for the expansion of the skill. Some of which could be listing issues with more than one filter, checking if there are dependency updates (such as from dependabots), and getting statistics on the team member's velocity in terms of contributions (number of commits/lines of code changes) over a time period. In addition, many participants actually mentioned that they didn't use GitHub in the industry, but rather, would prefer Robin's help on workflows for other platforms like Azure DevOps and Jira.

RQ3 To what extent is Robin feasible in supporting software developers?

From our study, we allowed our participants to freely compare the voice-assistant's completion of tasks on GitHub versus doing them manually to get their honest opinion on which they prefer. It is not surprising that the majority of the participants still preferred the web interface and doing tasks manually. The GitHub web interface allows our users to completely visualize everything that is happening and have all the information they need. In addition, it is probably a faster method as of now. Although Robin could help with the multi-tasking aspect of work, the manual way is still preferred even though it would require developers to switch context to do it themselves.

Robin needs to be more user tested and improved in terms of the design and the machine learning aspect. The design could be improved with more perspectives and manpower in the Robin implementation, but the machine learning aspect still requires more research and technological advances for Robin to be the assistant we imagined it to be. Robin still has quite a long way to go as of right now as the idea might be a little too advanced and premature for the time being.

Chapter 7

CONCLUSION

Throughout this thesis, we investigated the idea of having Robin, a GitHub-based voice-assistant, to help offload and automate common, repetitive software development tasks. Robin has the ability to assist users with issue-tracking and source code management tasks through the various intents it supports. Robin can perform an action or answer queries about a GitHub repository without the need for users to get into the context on their laptops.

In evaluating the effectiveness of having Robin in software development environments and teams, we conducted a small study on 10 participants. We recorded the number of attempts it took for the participants to get the correct answer from Robin in addition to collecting their reviews/opinions on a survey to see their perspectives. The main purpose of the study was to gain insights on the user experience as well as usability of our tool.

From the study, we found that many participants liked the idea of Robin, however, there are still many more enhancements that must be done. Robin is still a little premature in its field of voice-assistants. Although the current state of our project is usable, it doesn't give the user experience we prefer it to have. As shown in Chapter 6, there are still many limitations that should be addressed. In the following section, we will go over possible areas for future works.

7.1 Future Work

There are many directions and improvements that can be taken with Robin. Our current project described in this thesis is just the beginning.

One future work could be to add Robin support for different source code management tools and platforms such as Azure DevOps. Not all companies use GitHub, and by expanding coverage to different platforms, we are able to further spread the usage of Robin in workplaces. Another potential work could also be to have Robin fully implemented on Siri and Google Assistants as well. Not everyone will have an Alexa, so allowing other options such as Siri or the Google Assistant will provide users more access to Robin.

To address our limitations found in this thesis, a larger, longer, more controlled, and more realistic user experiment should be conducted. This is especially important as each participant can give us a new perspective on Robin in addition to more sample utterances as well. More user interaction can help gather more information and data as a sort of crowd-sourcing mechanism.

It is also imperative to improve our Robin conversation flow. Conversation flow is how a user would judge a voice-assistant. It makes or breaks the user experience. If an answer or response doesn't make sense, it turns the users off. One of the ways to help improve this aspect of Robin is to have more context and memory through the different intents in each session. For example, the user would not have to constantly state the issue or pull request number. Responding with concise and effective responses are key elements for a better user experience. Adding more confirmations for riskier tasks is also important, but we have to make sure we evaluate the trade-off between the extra time it takes to confirm an action versus our goal of productivity and efficiency in the software development workflow. In addition, we would ideally want Robin to understand usernames/aliases without having to spell it out. One idea is to possibly list all the collaborators of the repository.

In terms of access control, we should improve Robin to fully work with OAuth and allow each GitHub account to apply Robin actions/commands on the right user's behalf. It is not ideal to require the user *robin-teammate* be added to repositories and be recorded as the one performing actions. This diminishes the value of having a full log history for the repository of GitHub.

7.2 Discussion and Conclusion

From our research, we realize that our voice assistant might be a bit too premature as speech recognition and speech-to-text transcription are still very challenging tasks. A lot more research and iterations has to be done with more engineers to improve this idea of having a software development voice-assistant. With just voice, we have to balance the amount of information we provide to the user and whether that will result in increased or decreased productivity. As mentioned in [1], the second most distracting factor for developers is a "distracting work environment". Whether Robin adds to or reduces this distraction still remains to be determined.

Our study shows that some people support the idea of Robin if it were able to handle different situations and flow of conversation better. As of right now, there are a few intents that can be useful to our users such as assigning a user or a label to an issue. This project seems to be one of its first in the voice-assistant markets today as mentioned earlier in Chapter 2. Through Robin, we were able to set the foundation with our idea and the lessons that we learned. With more improvements, additions, and changes to our project, Robin has the potential to end up in many software engineering teams in the future.

As of right now it seemed that with our existing technology and implementations, having a similar, text-based assistant like Robin might be the best option. As mentioned in Chapter 3, chatbots help strip all the difficulties with speech recognition and speed-to-text conversion. We also would no longer be required to spell out usernames via voice any longer. In addition, as found in another research that is done with another student currently, it seems that a text-based GitHub provides more access control and authorization. By being able to encrypt the access token that was typed, the users are able to interact with the GitHub repositories through their own accounts.

Overall, a lot more research and experiments has to be done for our vision to be successful. The vision of being able to tell Robin to just handle the small tasks on our behalf as we continue to dig through our complex code bases is still very early on. However, with future improvements in our project and technology, we have high hopes that it will be possible very soon.

BIBLIOGRAPHY

[1] Stack Overflow Developer Survey 2019.
`https://insights.stackoverflow.com/survey/2019`.

[2] The State of the Octoverse. `https://octoverse.github.com/`.

[3] A. Abdellatif, K. Badran, and E. Shihab. Msrbot: Using bots to answer questions from software repositories. *Empirical Software Engineering*, 25(3):1834–1863, 2020.

[4] W. C. Bradford. Reaching the visual learner: teaching property through art. *The Law Teacher*, 11, 2004.

[5] N. Bradley, T. Fritz, and R. Holmes. Context-aware conversational developer assistants. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 993–1003. IEEE, 2018.

[6] Y. Chen, S. Oney, and W. S. Lasecki. Towards providing on-demand expert support for software developers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 3192–3203, New York, NY, USA, 2016. Association for Computing Machinery.

[7] B. da Silva, C. Hebert, A. Rawka, and S. Sereesathien. Robin: A voice controlled virtual teammate for software developers and teams. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 789–791, 2020.

[8] V. M. González and G. Mark. " constant, constant, multi-tasking craziness" managing multiple working spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, 2004.

[9] G. Mark, D. Gudith, and U. Klocke. The cost of interrupted work: more speed and stress. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 107–110, 2008.

[10] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz. The work life of developers: Activities, switches and perceived productivity. *IEEE Transactions on Software Engineering*, 43(12):1178–1193, 2017.

[11] P. Rodríguez, J. Markkula, M. Oivo, and K. Turula. Survey on agile and lean usage in finnish software industry. In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 139–148. IEEE, 2012.

[12] V. G. Stray, Y. Lindsjørn, and D. I. Sjøberg. Obstacles to efficient daily meetings in agile development projects: A case study. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 95–102. IEEE, 2013.

[13] V. G. Stray, N. B. Moe, and A. Aurum. Investigating daily team meetings in agile software projects. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 274–281. IEEE, 2012.