Northeastern University

CSYE 7230 Software Engineering Fall 2024

Project 2 Phase 1

Posted: 09/25/2024
Due: 10/06/2024, 11:59pm
Questions? email me at a.alali@northeastern.edu

Swishy!

My Assistant Manager (Swishy!) is a Swiss-knife secure chat bot that assist me and keep me organized. It helps me be productive in my day-to-day activities. Swishy understands a specific language, called the Swish language. Swishy! is a vault, a note taker, and a task manager. A user of Swishy! would need to get authenticated to be able to use Swishy!

High-level System Features

1. Users can activate the app functionalities using the Swish language

2. App users would register using their full name, password, and email. Then be able to login anytime afterwards to be allowed to use the full app features. Customers can reset their user password or delete themselves/their users. The initial user is a root admin and its password is root. Root can list and delete any user. All passwords are encrypted once saved.

```
> This is Swishy! if you want to learn how to talk to me, type @LearnSwissh
> @register @user @name Abdulkareem Alali @key a.alali@northeastern.edu
@password abc123
> @login @user @name Abdulkareem Alali @key a.alali@northeastern.edu
@password abc123
>
abd.alali@gmail.com is logged in
> @reset @user @key a.alali@northeastern.edu @password abc123
> @delete @user @key a.alali@northeastern.edu
```

3. Users creates a key and password pairs, they get saved and retrieved

```
> This is Swishy! if you want to learn how to talk to me, type @LearnSwissh
```

```
> save @key abd.alali@gmail.com @password abc123
> Can you @get me the @password for my email @key abd.alali@gmail.com
>
password123
> can you @save the email @key abd.alali@gmail.com using @password abc1234567
> can you @get me the @password for this @key abd.alali@gmail.com
>
abc1234567
```

4. Users creates a note, saves and retrieves them, notes are dated and timed

```
> This is Swishy! if you want to learn how to talk to me, type @LearnSwissh
> would you please @get me all the @notes from @date 3/3/2024
>
9/20/2024 13:23:34 the book clean code architecture seems to a great book to read, the
book author of the book is the famous uncle Bob.
9/20/2024 14:45:34 This is a great article
agileinaflash.blogspot.com/2009/02/first.html, great find!
> @save @note a weekend article https://dannorth.net/introducing-bdd/,
important topic! Will be a fun read
```

5. Users creates a task, saves and retrieves them, tasks are dated and timed, and a reminder can be scheduled, which is an email sent on a specific date, with the task embedded in the body, and the subject is “Swishy! reminds”

```
> This is Swishy! if you want to learn how to talk to me, say @LearnSwissh
> @save this @task submit the time estimates for the portfolio reskin UI work by the
end of this week.  And @remind me about it on @datetime 9/27/2024 13:00:00
> @save this @task grade project 1 by the end of Tuesday @remind @datetime 9/27/2024
13:00:00
> @get me all the @tasks from @date 9/20/2024
>
9/20/2024 13:23:34 submit the time estimates for the portfolio reskin UI work by the
end of this week.
9/20/2024 14:45:34 grade project 1 by the end of Tuesday
```

6. If the user types `@LearnSwissh`, app will display a tutorial on how to speak and interact with the chat bot Swishy!

GitHub

3

Use your GitHub account, and if you don't have one, make one, then:

1. Create a new repository and call it **Swishy**. Make it private, and add the team to it aalali, and your classmate GitHub account: cecillyaliu or snguyen1122
2. Create a branch, call it **swishy-phase1-feature**
3. Open a draft pull request, then commit your code as frequently as possible
4. Once finished, add us as reviewers (due date is at this point)
5. We will review, and add comments, and then we will approve
6. You merge into the main, delete the feature branch

You might find this reference and this reference helpful.

Inception Phase Artifacts to Submit

4

For this assignment, proceed through the inception phase and develop artifacts. Use Larman's e/3 book format, tables, categorization, material, context and definitions **closely**. Artifacts to submit are:

A. Requirements

1. Vision (`vision.txt`) including Introduction, Positioning, Stakeholder Descriptions, Product Overview, and Summary of System Features.

2. Use-Case Model

   a. All use cases are named and written in brief format and only 10% in fully dressed format and deeply investigated. This 10% will be the most architecturally important, risky, and of high-business value. Each use case should be in a separate text file. Name each use case file using the use case name (`uc01_USECASENAME.txt`) e.g. `usecase_01_process_order.txt`

   b. Use case diagrams (`usecase_diagram_v1.png`) are drawn using a UML tool of choice

3. Supplementary Specification (`supplementary_specification.txt`) including URPS+ which are Usability, Reliability, Performance, Supportability, + Constraints (Implementation, Interface, Operations, Packaging, Legal)[1]

4. Glossary/Data Dictionary (`glossary.txt`)

5. Business (domain) Rules (`business_rules.txt`)

[1] If a category of an artifact doesn't apply to our system, then, there is no need to be used. If a new category is needed UP is flexible and that would be acceptable.

Notes

I. For any text (`TXT`) or source (`Java`) files, use a header (example below):

```
/*-----------------------------------------------------------------

        App: Swishy!
        Artifact Name: xxx.java
        Create Date: Jan 30, 2100
        Author: Abdulkareem Alali, a.alali@northeastern.edu
        Version: 1.0

-----------------------------------------------------------------*/
```

II. For PNG files, include the version number into the file name.

III. Use draw.io, UMLet, UMLetino, yUML for UML diagraming or any UML tools of your choice. More dedicated software such as Lucidcharts or Microsoft Visio can also be used. Don't use hand drawing or something as simple as Microsoft Paint.

IV. All diagrams must be submitted in PNG type files. For documents, use TXT type files.

V. Make multiple commits with adequate messages to show your process/progress.

VI. Commit all your artifacts into your project 2 repository, each should be placed in the right folder as in the directory tree structure, see next point

VII. Your directory tree structure **should** (exact names and structure) look like:

```
.

|-- swishy
|   |-- Requirements
|   |   |-- UseCaseModel
|   |   |   |-- uc_diagram.png
|   |   |   |-- uc01_USECASENAME.txt
|   |   |   |-- uc02_USECASENAME.txt
|   |   |   |-- ...
|   |   |-- vision.txt
|   |   |-- supplementary_specification.txt
|   |   |-- glossary.txt
|   |   |-- business_rules.txt
```

Grading

The grade will be broken down as follows:

Point	Mark
1.	5/100
2. a.	50/100
2. b.	20/100
3.	15/100
4.	5/100
5.	5/100

References

- Akronzoo.org
- Nationalzoo.si.edu

Use Cases (UC), Features (F), and Actors Table (Instructor's Vision/Expectation)[2] [3]

Primary Actors	Goal	UC#	F#	To Implement	Most Critical
App User	Manage User	1		Yes	*
	Manage Tasks	2		Yes	
	User Authentication		1	Yes	
	Manage Notes	3		Yes	
	Manage Tasks	4		Yes	
	Manage Vault	5		Yes	*
	Logging		2	Yes	
Root User	Manage Root	6		Yes	
	User Authentication		3	Yes	
	Manage Users	7		Yes	
	Logging		4	Yes	

[2] This can be enhanced, and it's open to evolve. There is **no** need to commit and follow this table, but feel free to do so. This is an **example** that breaks down **Swishy**! to use cases and features

[3] Non-primary actors have not been included