WASHINGTON STATE UNIVERSITY
VANCOUVER

CS 320 Course Project Final Report

for

Python Onboarding For Incoming Students (POFIS)

Version 0.1

Prepared by

Group Name: Saab

Rebecca Daniel	11585986	rebecca.daniel@wsu.edu
Andrew Cornish	11571297	andrew.cornish@wsu.edu
Samuel Dunn	11453733	samuel.i.dunn@wsu.edu
Abdi Vicenciodelmoral	11554779	a.vicenciodelmoral@wsu.edu

Date: 12/13/2019

Contents

FIGURES

1 Introduction

1.1 Project Overview

POFIS aims to become a standalone educational assistance asset for WSU-V. Its primary target audience is incoming transfer students with exposure to programming, but a lack of experience with the Python programming language. POFIS will contain useful reference pages, code snippets, links to established python modules, and an interactive tutorial. While it will be self-sufficient, it is not a substitute for education, and therefore will not aim to teach programming, merely standards and best practices for Python. Instructors may refer students to POFIS if they feel a student would benefit from the assets contained within POFIS.

1.2 Definitions, Acronyms and Abbreviations

POFIS	Python Onboarding For Incoming Students
WSGI	Web Server Gateway Interface
WSU-V	Washington State University, Vancouver

1.2 References and Acknowledgments

IEEE Guide to Software Requirements Specifications, 1st ed., The Institute of Electrical and Electronics Engineers, Inc, New York, NY, 1984, pp.1-26

2 Design

2.1 System Modeling

During the implementation of POFIS, some modifications were necessary from the previously anticipated system modeling. Furthermore, it was determined that the following diagram provided a better representation of the process in a more concise manner.

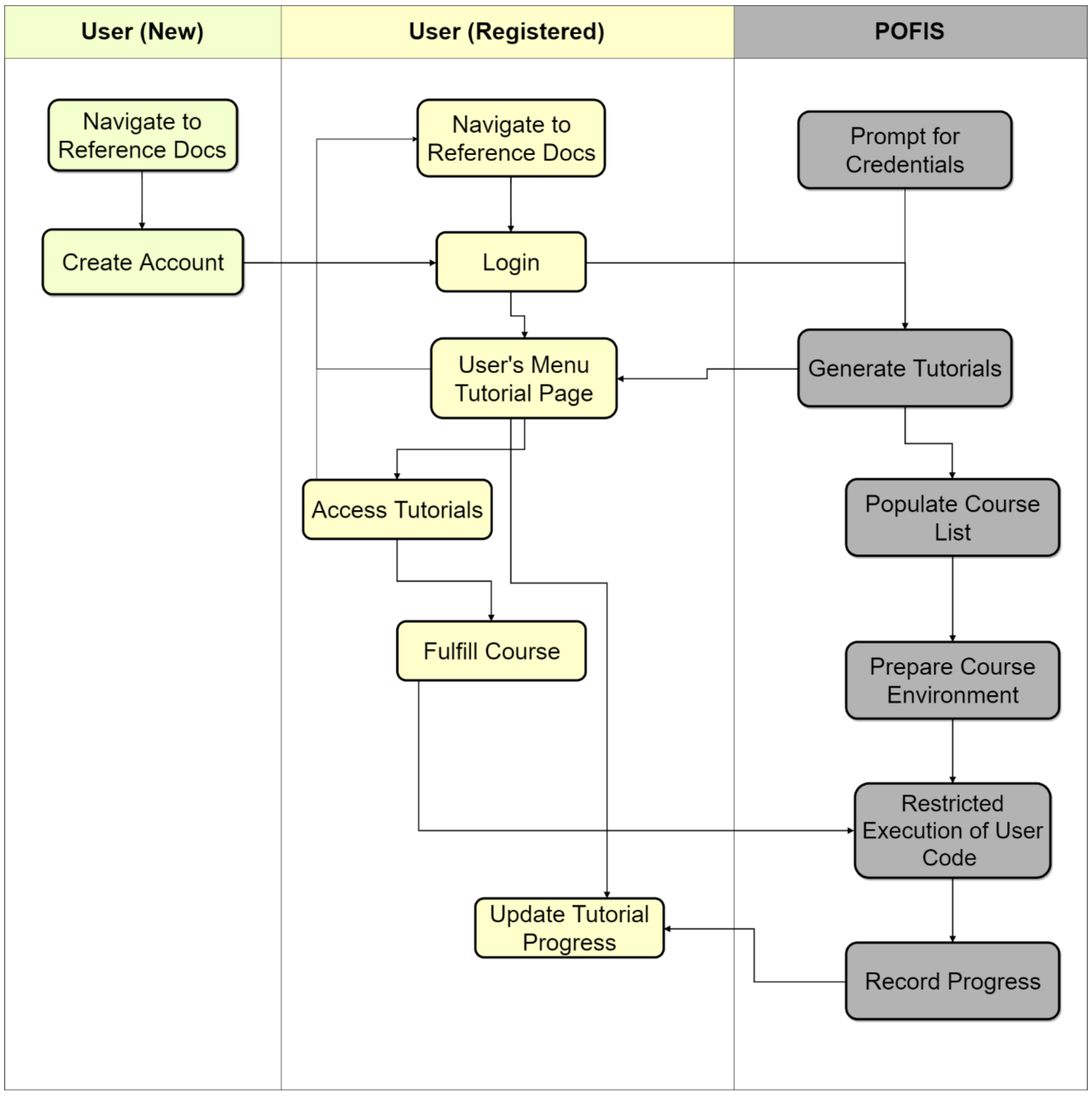
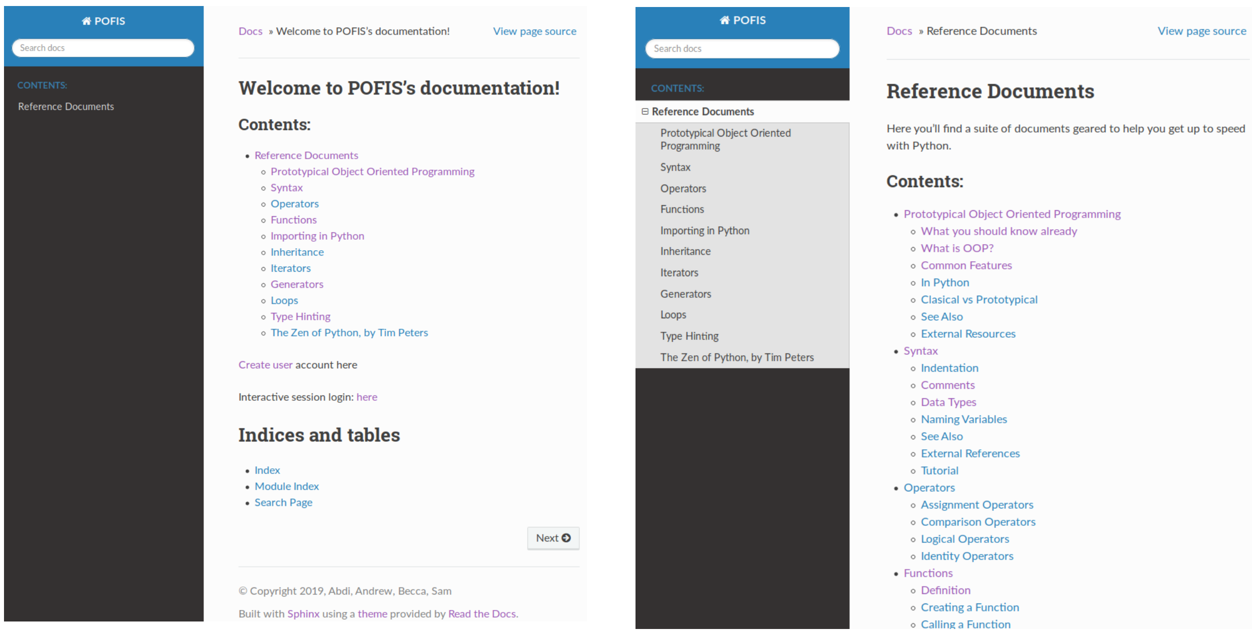


Figure 1: POFIS System Diagram

2.2 Interface Design

The following screenshots are representative of subsystems within POFIS.



POFIS

Search docs

CONTENTS:

Reference Documents

Docs » Welcome to POFIS's documentation! View page source

Welcome to POFIS's documentation!

Contents:

- Reference Documents
  - Prototypical Object Oriented Programming
  - Syntax
  - Operators
  - Functions
  - Importing in Python
  - Inheritance
  - Iterators
  - Generators
  - Loops
  - Type Hinting
  - The Zen of Python, by Tim Peters

Create user account here

Interactive session login: here

Indices and tables

- Index
- Module Index
- Search Page

Next

© Copyright 2019, Abdi, Andrew, Becca, Sam

Built with Sphinx using a theme provided by Read the Docs.

POFIS

Search docs

CONTENTS:

Reference Documents

Prototypical Object Oriented Programming

Syntax

Operators

Functions

Importing in Python

Inheritance

Iterators

Generators

Loops

Type Hinting

The Zen of Python, by Tim Peters

Docs » Reference Documents View page source

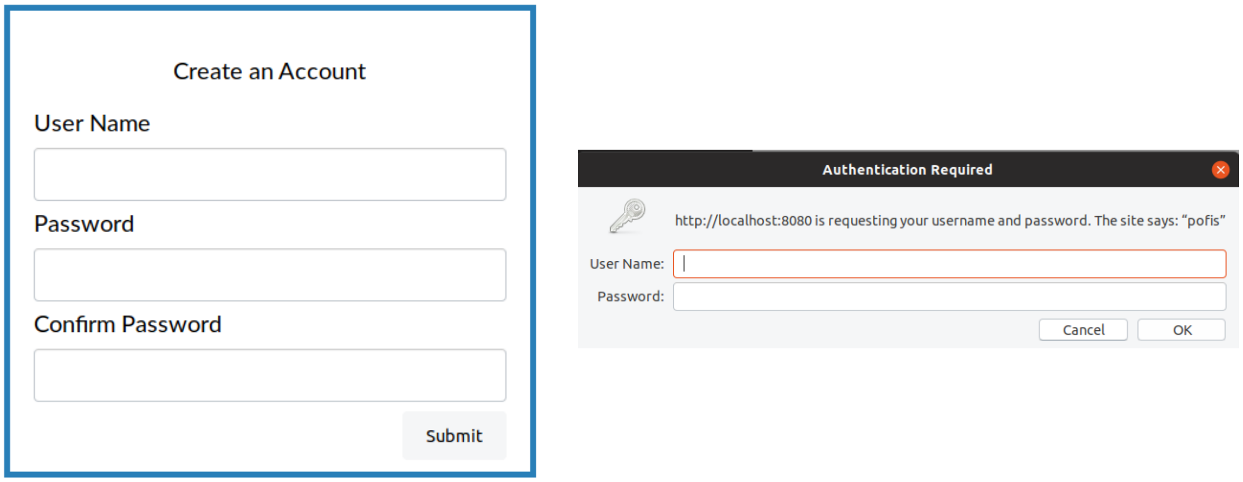
Reference Documents

Here you'll find a suite of documents geared to help you get up to speed with Python.

Contents:

- Prototypical Object Oriented Programming
  - What you should know already
  - What is OOP?
  - Common Features
  - In Python
  - Clasical vs Prototypical
  - See Also
  - External Resources
- Syntax
  - Indentation
  - Comments
  - Data Types
  - Naming Variables
  - See Also
  - External References
  - Tutorial
- Operators
  - Assignment Operators
  - Comparison Operators
  - Logical Operators
  - Identity Operators
- Functions
  - Definition
  - Creating a Function
  - Calling a Function

Figure 2: Home and References page examples



Create an Account

User Name

Password

Confirm Password

Submit

Authentication Required

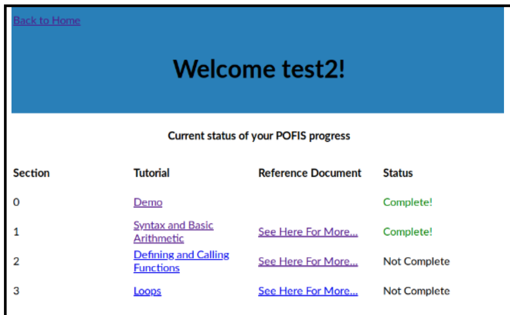
http://localhost:8080 is requesting your username and password. The site says: "pofis"

User Name:

Password:

Cancel OK

Figure 3: Account creation and login screens

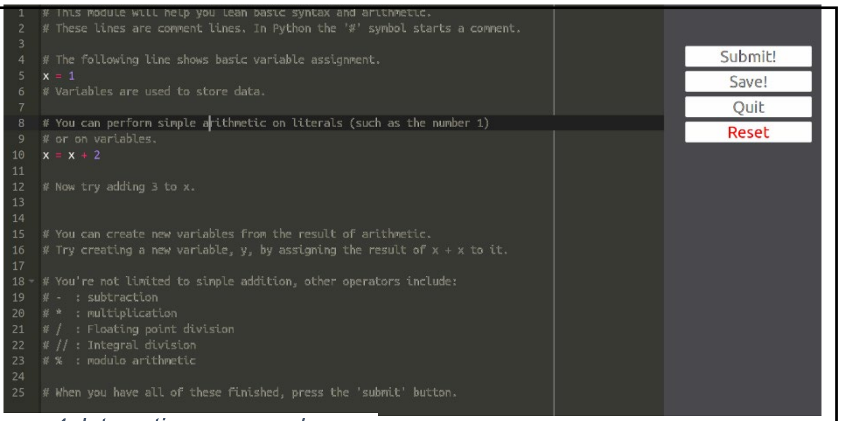


Back to Home

Welcome test2!

Current status of your POFIS progress

Section	Tutorial	Reference Document	Status
0	Demo		Complete!
1	Syntax and Basic Arithmetic	See Here For More...	Complete!
2	Defining and Calling Functions	See Here For More...	Not Complete
3	Loops	See Here For More...	Not Complete



```
# This module will help you lean basic syntax and arithmetic.
# These lines are comment lines. In Python the '#' symbol starts a comment.

# The following line shows basic variable assignment.
x = 1
# Variables are used to store data.

# You can perform simple arithmetic on literals (such as the number 1)
# or on variables.
x = x + 2

# Now try adding 3 to x.


# You can create new variables from the result of arithmetic.
# Try creating a new variable, y, by assigning the result of x + x to it.

# You're not limited to simple addition, other operators include:
# -  : subtraction
# *  : multiplication
# /  : Floating point division
# // : Integral division
# %  : modulo arithmetic

# When you have all of these finished, press the 'submit' button.
```

Figure 4: Interactive menu and interface

3 Implementation

3.1 Development Environment

The specific development environment varied throughout the implementation of POFIS. Most commonly, Ubuntu was utilized as the preferred operating system. VSCode and Sublime3 were both used for code editing. Languages, Frameworks and API's that were used are as followed; Python, JavaScript, HTML, CSS, Jinja2 and Sphinx. In addition, GitHub was used extensively for collaboration and project version control.

3.2 Task Distribution

Throughout the design and implementation process, all team members contributed to every aspect of the system. The following table highlights the specific tasks which were spearheaded by each member.

Rebecca Daniel	Authored Sphinx restructured text pages and create user account page
Andrew Cornish	Report editor, Authored Sphinx restructured text pages
Samuel Dunn	Sphinx and tutorial code editor implementation, mentor
Abdi Vicenciodelmoral	UML diagram designer, authored tutorial splash page

3.3 Challenges

As the project progressed, the team faced many challenges. The most notable challenge faced was safely executing user generated code.

3.3.1 Safe execution of user generated code

It was decided that pure lexical analysis of user input would be prone to logical gaps and errors. To avoid this, it was determined that it was necessary to execute user generated code. The decision to execute user-generated code posed many security threats. For instance, malicious users could gain access to the servers file system and potentially delete files or execute malicious code. In order to remove this security risk, it was determined that the user generated code must be executed in a controlled environment. This was accomplished in POFIS by substituting the interpreters core functionality with unittest.mock. MagicMock instances, which will prevent the user from performing actions such as, opening files, importing modules, reading from stdin, and other actions deemed unsafe.

3.3.2 Infinite loop

Although the most hazardous functionality a user could access was mitigated by the previous solution, there remained a method which the user could submit code which could cause an infinite loop. If a user, maliciously or accidently, submitted code which caused an infinite loop the result could render the service inoperable. To mitigate this possibility, user generated code is evaluated in a separate, killable, process from the webserver. The webserver will periodically check for completion of execution and evaluation and reply to the user with results when they are made available. If the execution time of user generated code exceeds 500ms, the server will kill the evaluation process and inform the user of the timeout failure.

4 Testing

4.1 Testing Plan

User account creation: The user account creation shall be tested with multiple accounts using various length names. Attempts will be made to ensure only the proper syntax is allowed in each field.

User login: User login shall be tested using multiple accounts.

Home page: Ensure the home page works on listed browsers.

Tutorial pages: Multiple pages will be tested for functionality. As a standard practice, each addition to a tutorial will be fully tested prior to implementation on the live server

4.2 Test for Functional Requirements

User account creation: Checked if username already existed, if password entered was the same in both password and confirm password field and followed minimum password requirements. Minimum requirements are; length must be between 6 and 10 and include 1 digit.

Tested tutorials: Tested multiple accounts and various tutorial links to verify the tracking of completion was correct.

User submitted code: Verify code correctness. Also ensure test completion did not happen unless entire tutorial was completed.

Link “back to home” worked on create account and tutorial page. Verify links to from POFIS to account creation and tutorial page works, and links from tutorial page to their respective reference pages were correct.

4.3 Test for Non-functional Requirements

The following non-functional requirements were tested thoroughly without an instance of failure.

1. Home and tutorial selection pages shall not take longer than 15 seconds to load.
2. User account creation shall not take longer than 10 seconds to confirm or refuse.
3. Initial session loading of any tutorial shall not take longer than 15 seconds.
4. Further steps of a tutorial should not take longer than 5 seconds to load.
5. Code submitted by a user shall take no longer than 10 seconds to receive feedback.
    a. Exceptions to this rule may occur if tutorial has extensive coding involved.
    b. If an exception is warranted, the instructions must indicate that the processing time may be longer than the norm.

4.4 Safety and Security Requirements

The following guidelines were implemented and have been tested thoroughly without error.

1. Password must be 6 to 10 characters in length
2. Passwords must contain at least one numeric character

4.5 Hardware and Software Requirements

Tests shall be performed using Windows, Macintosh and Linux operating systems using common web browsers for the various systems. If possible, the listed operating systems shall be tested using Mozilla Firefox v.68+ and Google Chrome v.7+. Furthermore, the Windows system will be further tested using Microsoft Edge v.44+, and tests using, Macintosh Safari v.11+ will be conducted on the Macintosh operating system.

5 Analysis

Milestone 1: The entire POFIS team spent roughly 3-4 hours together on the SRS documentation.

Milestone 2:

Abdi: 3 hours on creating diagrams.

Andrew: 3-4 hours on documentation overview

Sam: 2 hours of Git creation and design of code project structure

Becca: 3 hours Documentation / diagram creation and review

Milestone 3:

This milestone took the most work due to a high learning curve and exhaustive testing

Sam: 6-7 hours on project set up, functionality, code review, and mentoring

Andrew: 5-6 hours document creation, creation of reference documents

Abdi: 5-6 hours of creation of diagrams, reference docs, splash page for tutorial page

Becca: 5-6 hours of creation of reference docs, account creation page, 401 redirect page

6 Conclusion

Project management collaboration with the use of Github. This allowed us to become more familiar with pull requests, resolving merge conflicts, and git command syntax for the terminal.

We also learned how to integrate cherrypy with Sphnix using their Read the Docs template. We also used Jinja2 for HTML page creation.

This project reinforced our base knowledge of Python, HTML, and CSS while helping us improve on logical condition when testing.

Appendix A – Group Log

11/5:

- Discussed project set up. Tutorial about setting up virtual environments and how to create the references pages for Sphnix

11/24:

- Continue to work on reference docs. Implemented the code editor within the project.

11/28:

- Tested localhost:8080 on everyone's device to ensure proper set up.

12/2:

- Started Milestone3 final report. Gave out assignments for the various sections.

12/10:

- Creation of create user page, tutorial splash page, tutorials. Final clean-up of overall design and feel of POFIS.

12/12:

- Presentation day! Forgot to talk about EC, we used Factory Pattern.
- Finish working on Milestone3 Final Report
- Turn in!