Roger Johnson

ePortfolio

Artifact Introduction

I chose to build off from a singular artifact that was created during the Computer Science program. Specifically, from CS 410 Software Reverse Engineering where I was tasked to extract assembly code from an object file and convert it into C. Selecting a single artifact allows me to showcase the knowledge, skills, and abilities gained from my time at Southern New Hampshire University. To begin I enhanced the C code by transferring it into Python, then added needed features, and finally implementing a real-time database.

Code Review

The video below shows a code review performed on the original artifact. The review covers the origins of the artifact, existing functionality, code analysis, and the early thought process of potential enhancements.

Code review

Artifact 1: C to Python

Category: Software Design and Engineering

pre-enhancement/post-enhancment

The original artifact comes from CS-410: Software Reverse Engineering where we were tasked on extracting assembly code from an object file and converting it into C. To complete that assignment, I used the terminal command “objdump” to extract the assembly code from the object file and then proceeded to convert it into C. The program was designed to keep track of 5 student’s grades. During the conversion from assembly, I kept the functionality of the program exact, including all flaws.

I chose this artifact for the software design and engineering category because it demonstrates my ability to utilize reverse engineering tools and practices, knowledge of assembly language, and the skill to convert between multiple languages.

To accomplish that I converted the C code into Python and show the code below, split up into respective sections.

```
const char * STUDENTS[5] = { "Jim", "Tom", "Ben", "Alice", "Ruby" };
char GRADES[6] =  {'A','C','C','D','F','\0'};

STUDENTS = ["Jim", "Tom", "Ben", "Alice", "Ruby"]
GRADES =  ['A','C','C','D','F','\0']
```

```
int ReadUserInfo(){
  int password = 0;
  char str1[15];
  puts("Enter name:");
  scanf("%s", str1);
  puts("Enter password:");
  scanf(" %d", &password);
  return password;
}

def ReadUserInfo():
    name = input("Enter name:")
    password = input("Enter password:")
    return password
```

```
bool CheckUserPermissionAccess(int password){
  bool check = 0;
  if(password == 123){
    check = 1;
  }
  else{
    check = 0;
  }
  return check;
}

def CheckUserPermissionAccess(password):
    if int(password) == 123:
        check = True
    else:
        check = False
    return check
```

```
void DisplayStudentInformation(){
  int i;
  for(i = 0; i <= 4; ++i){
    printf("%s %c\n", STUDENTS[i], GRADES[i]);
  }
}

def DisplayStudentInformation():
    for index, name in enumerate(STUDENTS):
        print(name + ": " + GRADES[index])
```

```
int main()
{
  int password;
  int i;
  char choice;
  password = ReadUserInfo();
  if(CheckUserPermissionAccess(password) == 1){
    puts("Welcome professor. Below are all student grades");
    DisplayStudentInformation();
    puts("Adjust grades for students?");
    scanf(" %c", &choice);
    if(choice == 'Y'){
      printf("Enter the GPA for students one at a time\n");
        for(i=0; i <= 4; ++i){
          printf("%s", STUDENTS[i]);
          scanf(" %c", &GRADES[i]);
        }
      puts("You have successfully updated class grades. The grades are now as
follows:");
      DisplayStudentInformation();
    }
  }
  return 0;
}

password = ReadUserInfo()
if CheckUserPermissionAccess(password) == True:
    print("Welcome professor. Below are all student grades")
    DisplayStudentInformation()
    if input("Adjust grades for students? Y/N:") == "Y":
        print("Enter the GPA for students one at a time")
        for index, name in enumerate(STUDENTS):
            GRADES[index] = input(name + ":")
        print("You have successfully updated class grades. The grades are now as
follows:")
        DisplayStudentInformation()
```

Converting the C code into Python isn't extremely complicated as long as you have a background in programming and understand the syntax. Because this was one of my first experiences with Python, I had to learn the proper way to take an input from the user and how indices are gathered from a for-loop. Since the original code had no form of documentation, after converting to Python I went through and added the needed comments and code documentation that can be seen here.

Artifact 2: Sign-in and Input Handling

Category: Algorithms and Data Structure

pre-enhancement/post-enhancment

This artifact is the above finished Python code. While the program is running, it allows anyone that knows the password to view and change the grades of five students.

I chose to build off from my first artifact for the algorithms and data structure category because of how much room there was for improvement in the code. Before my enhancements, the program didn't have a secure sign-on, it didn't use the entered username for any checks and also used '123' as a password for everyone. The program also did not care what was entered and would exit on certain inputs.

Below I go into the main enhancements that were made.

For the sign-in feature, I used the existing function ReadUserInfo() to gather both username and password. To accommodate a more secure system, when the user is entering their password, the text will not be shown. Since I am using lists to store the credentials of the teachers and their passwords, the indices of the username and password together act as a key-value pair. So to check if the entered values match, I used a for-loop with enumerate to track the index and compare both lists.

```python
def SignInFeature():
    (password, name) = ReadUserInfo()
    for index, teacher in enumerate(TEACHERS):
        if name == teacher and CheckUserPermissionAccess(index, password) == True:
            signin = True
            break
        else:
            signin = False
    return signin
```

For the input handling I made a function to ensure that the program only continues or exits on the specific value. To do this the function has 2 parameters: output, restriction. The output is the string that prompts the user for input. The restriction is what the user is allowed to enter, usually a list. The function starts with an infinite loop that exits on return. It prompts the user for their input and goes into a try-except clause. This ensures that the user is entering the desired variable type. Next it goes into an if statement that will only be true if the value entered matches any value within the restriction parameter.

```
def HandleInput(output, restriction):
    while True:
        value = input(output)
        try:
            if type(restriction[0])(value) in restriction:
                return value
            else:
                print("INVALID INPUT. TRY AGAIN")
        except:
            print("INVALID INPUT. TRY AGAIN")
```

Code documentation can be found here.

As stated in the last category, I don't have much experience in Python but I am starting to learn that it's a very intuitive language and is easy to pick up with the knowledge I already have. I learned how to type cast using a variables type we already know. One challenge I had was when I started the type casting I was getting an error if the restriction I was looking for didn't match the entered type. I fixed this with a try-except clause which catches the error and notifies the user that the input was incorrect.

Artifact 3: Firebase Integration

Category: Databases

pre-enhancement/post-enhancment

The final artifact will be a continuation of the last two categories where I took C code, converted it into Python and then added greater functionality.

I chose this artifact for the database category to demonstrate my ability to utilize APIs and implement a real-time database. Before the enhancements, the code was using lists that would revert on program close. It also relied on indexing those lists and hoping the needed matches were present. The code itself had plain text passwords within a list which was a security risk.

Below I go through the process in implementing Firebase within Python.

In order to get the Firebase API working in Python, a module called Pyrebase is needed. To configure Pyrebase, it needs to know where to connect with Firebase using sensitive data. In an attempt to separate that data from the program itself I created a json file to hold it all. When the program starts, Firebase is initialized and the proper connections are made.

```
def FirebaseSetup():
    with open('cred.json') as f:
        config = json.load(f)
    return pyrebase.initialize_app(config)
```

Firebase allows you to import a json file of the data you want to track so in order to keep the structure of the data in the original artifact, I took the lists and created the json.

```
{
  "USER": {
    "Admin": {
      "name": "Admin",
      "password": "password"
    },
    "Johnson": {
      "name": "Roger Johnson",
      "password": "Roger"
    }
  },
  "STUDENTS": {
    "Jim": {
      "grade": "A"
    },
    "Tom": {
      "grade": "C"
    },
    "Ben": {
      "grade": "C"
    },
    "Alice": {
      "grade": "D"
    },
    "Ruby": {
      "grade": "F"
    }
  }
}
```

Now that the data is being stored within Firebase and the connection is setup, I can start to utilize the API. To keep the code organized and to keep the main code block clean, I made a new function to change the student's grades. In order to change every student's grade one at a time, we need a for-loop. In order to loop through we need a list of just student names so we access the database STUDENTS and get() the value of the first key. Finally for each student we can change individual grades. To change an existing entry we use update() so an example would look like: Database.child("STUDENTS").child("Jim").update({"grade" : "A"}). Since we are looking for input from the user though the code is a little more complicated than that when input handling is thrown in.

```
  def ChangeStudentGrades(database):
      print("Enter new grade for students one at a time")
      STUDENTS = database.child("STUDENTS").get()
      for student in STUDENTS.each():
          database.child("STUDENTS").child(student.key()).update({"grade":
              HandleInput(student.key() + ":",
['a','A','b','B','c','C','d','D','f','F']).upper()})
```

Since we no longer have the lists within the code, there must be modifications made to the existing functions. In order to display the student information from the database, there needs to be a for-loop like the one in ChangeStudentGrades(). Just this no values are changing.

```
  def DisplayStudentInformation(database):
      STUDENTS = database.child("STUDENTS").get()
      for student in STUDENTS.each():
          print(student.key() + ": " +
str(database.child("STUDENTS").child(student.key()).child("grade").get().val()))
```

Then for the sign-in feature, the If statement changes to compare the user entered name and password with the correct key-value pair within the Firebase database.

```
  def SignInFeature(database):
      (password, name) = ReadUserInfo()
      if database.child("USER").child(str(name)).child("password").get().val() ==
password:
          signin = True
      else:
          signin = False
      return signin
```

The main code block of the program doesn't look like much but gets a lot done in presenting the information to the user.

```
  firebase = FirebaseSetup()
  db = firebase.database()

  while SignInFeature(db) != True:
      print("Incorrect credentials. Try again")

  print("Welcome professor. Below are all student grades")
  DisplayStudentInformation(db)

  choice = HandleInput("Adjust grades for students? Y/N:", ['Y','y','N','n'])
  if choice == 'Y' or choice == 'y':
      ChangeStudentGrades(db)
      print("You have successfully updated class grades. The grades are now as
follows:")
      DisplayStudentInformation(db)

  print("Goodbye")
```

Code documentation can be found here.

The integration of the Firebase real-time database turns this simple console program that had no way to save information into a crude full-stack system. Albeit the program still runs in the console, it now keeps all grades up to date and uses simple security measures to limit access to just the teachers. After taking a program this far with a language I wasn't familiar with I feel confident in my abilities as a software engineer.