# Requirement Analysis – Jomas Benfell (20240825)

## Objective

The objective is to develop a Help Desk ticketing system prototype for handling tickets from internal customers within the organization. The system should allow submitting, tracking, and resolving tickets created by staff members seeking assistance from the Help Desk.

## Function Requirements

## Ticket Submission

- Users should be able to submit tickets by providing the following information:
    - Staff ID
    - Ticket Creator Name
    - Contact Email
    - Description of the Issue

## Ticket Response

- Users should be able to respond to tickets, with the default response on new tickets being “Not Yet Provided”.

## Changing Passwords

- If the ticket description has “password change” in it (regardless of upper or lower case), the system should automatically generate a new password using the following:
    - The first two characters of the staffID, followed by the first three characters of the staff members name.

## Tracking Tickets

- The system should show the number of tickets created, needing solved, and being resolved.
- It should always show these in the form of statistics in the main menu.

## Displaying Ticket Information

- Users should be able to view ticket information, including the ticket number, staff member name, email address, description, response, and ticket status.

## Ticket Interaction

- Users should be able to interact with tickets in the way of creation, reopening, or reclosing tickets, as well as response as previously mentioned.

## Not Functional Requirements:

### Performance:

- There should be minimal delay between the users' input and the system responding, reading, and writing the necessary data required.

### Reliability:

- The system should be reliable, with the code being able to accommodate for any unexpected inputs.
- The system should be able to accurately track ticket information and statuses without causing any errors.

### Security:

- In the final product, all user data should be stored securely, and protected from unauthorized access by encryption. As this is a prototype, I have elected to not include this. However, for the final product the data stored within the spreadsheet would be within a secured database where the data would be encrypted.

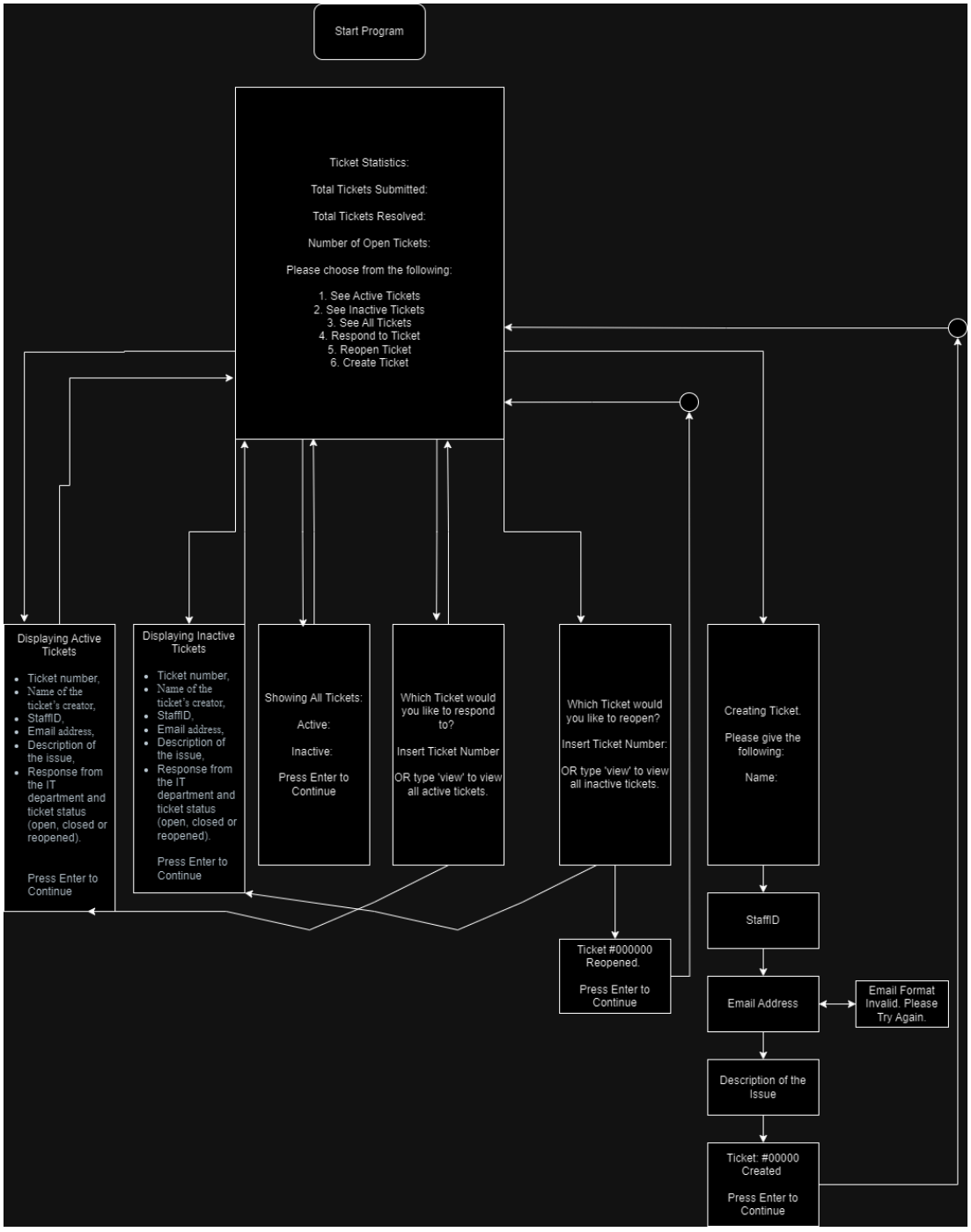
- In the final product, the passwords generated should also have proper security standards, such as upper case and lower-case characters, as well as numbers and symbols. However, for the prototype they will be generated using the first two characters of the Staff ID followed by the first three characters of the staff members' name, as requested by the client.

## Usability:

- The system must feature a user-friendly interface, making it easy to understand and use.
- For the prototype, this is all ran via command prompt on Windows. However, for a final product this would likely be web-based.

## Design:

- The system will be structured using classes and functions to allow for multiple different functionalities as requested by the client. The initial code structure will include classes for managing tickets, as well as functions for submitting tickets, submitting responses, displaying tickets, creating tickets, as well as opening/closing tickets. Additionally, I have created a flowchart below to show the systems flow.



## Developing the Solution - Explanation of the Code

As you will see the code below, I wish to give some explanation to parts of it.

```
import os
import csv

# Ticket data for example output to put inside of a csv file.
tickets_data = [
  {
    'Ticket Number': 2001,
    'Staff Name': 'Inna',
    'Staff ID': 'INNAM',
    'Email': 'inna@whitecliffe.co.nz',
    'Description': 'My monitor stopped working',
    'Response': 'Not Yet Provided',
    'Status': 'Open'
  },
  {
    'Ticket Number': 2002,
    'Staff Name': 'Maria',
    'Staff ID': 'MARIAH',
    'Email': 'maria@whitecliffe.co.nz',
    'Description': 'Request for a videocamera to conduct webinars',
    'Response': 'Not Yet Provided',
    'Status': 'Open'
  },
  {
    'Ticket Number': 2003,
    'Staff Name': 'John',
    'Staff ID': 'JOHNS',
```

```
        'Email': 'john@whitecliffe.co.nz',
        'Description': 'Password change',
        'Response': 'New password generated: JOJoh',
        'Status': 'Closed'
    }
]

# Above data gets placed into a .CSV file if file doesn't already exist, to retain any/all changes and data of the database.
filename = 'database.csv'
with open(filename, 'w', newline='') as file:
    database = csv.DictWriter(file, fieldnames=['Ticket Number', 'Staff Name', 'Staff ID', 'Email', 'Description', 'Response', 'Status'])
    database.writeheader()
    database.writerows(tickets_data)

print(f"Ticket Database created: {filename}.")



class Ticket:
    total_tickets_submitted = 0 # 0 by default, changes based on .csv file
    total_tickets_resolved = 0 # 0 by default, changes based on .csv file

    def __init__(self, ticket_id, staff_name, staff_id, email, description, response=None, status="Open"): # Response is none by default. Status Open by default.
        self.ticket_id = ticket_id # 2000 + 1 over the last ticket (e.g if last ticket is 2002, next one will be 2003.)
        self.staff_name = staff_name
        self.staff_id = staff_id # e.g. INNAM, MARIAH, etc.
        self.email = email # Requires @ to be used or won't submit, code on that below at elif choice == 6.
        self.description = description # The Issue
        self.response = response if response is not None else "Not Yet Provided"
```

```
        self.status = status

    @classmethod
    def load_tickets(cls, filename):
        tickets = []
        with open(filename, 'r', newline='') as file:
            reader = csv.DictReader(file)
            for row in reader:
                ticket = Ticket(
                    int(row['Ticket Number']),
                    row['Staff Name'],
                    row['Staff ID'],
                    row['Email'],
                    row['Description'],
                    row.get('Response'),  # No default value for response
                    row.get('Status', 'Open')  # Default to 'Open' if status is not provided
                )
                tickets.append(ticket)
        return tickets

    @classmethod
    def save_tickets(cls, tickets, filename):
        with open(filename, 'w', newline='') as file:
            writer = csv.DictWriter(file, fieldnames=['Ticket Number', 'Staff Name', 'Staff ID', 'Email',
'Description', 'Response', 'Status'])
            writer.writeheader()
            for ticket in tickets:
                writer.writerow({
                    'Ticket Number': ticket.ticket_id,
                    'Staff Name': ticket.staff_name,
                    'Staff ID': ticket.staff_id,
```

```
            'Email': ticket.email,
            'Description': ticket.description,
            'Response': ticket.response,
            'Status': ticket.status
        })



def respond_to_ticket(tickets, ticket_id):
    for ticket in tickets:
        if ticket.ticket_id == ticket_id:
            response = input("Enter your response: ")
            ticket.response = response
            ticket.status = "Closed"
            print("Response Submitted. The Ticket is now Closed.")
            return True
    print("TicketID not found.")
    return False



def reopen_ticket(tickets, ticket_id):
    for ticket in tickets:
        if ticket.ticket_id == ticket_id:
            ticket.status = "Open"
            print("Ticket reopened successfully.")
            return True
    print("TicketID not found.")
    return False



def TicketStats(tickets):
```

```
    print("\nStatistics:")
    print(f"Tickets Created: {len(tickets)}")
    print(f"Tickets Resolved: {sum(ticket.status == 'Closed' for ticket in tickets)}")
    print(f"Tickets to Solve: {len(tickets) - sum(ticket.status == 'Closed' for ticket in tickets)}")


def show_tickets(tickets):
    for ticket in tickets:
        print("Ticket Number:", ticket.ticket_id)
        print("Ticket Staff:", ticket.staff_name)
        print("Staff ID:", ticket.staff_id)
        print("Email Address:", ticket.email)
        print("Description:", ticket.description)
        print("Response:", ticket.response)
        print("Ticket Status:", ticket.status)
        print()


def show_main_menu():
    print("\nPlease Choose from the Following:")
    print("1. See Open Tickets")
    print("2. See Closed Tickets")
    print("3. See All Tickets")
    print("4. Respond to Ticket")
    print("5. Reopen Ticket")
    print("6. Create Ticket")
    print("7. Exit")


def main():
    filename = 'database.csv'
```

```
tickets = Ticket.load_tickets(filename)

while True:
    os.system('cls' if os.name == 'nt' else 'clear')  # Clear the screen
    TicketStats(tickets)
    show_main_menu()

    choice = input("\nEnter your choice: ")  # Add an extra newline before the prompt

    if choice == '1': # See Open Tickets
        open_tickets = [ticket for ticket in tickets if ticket.status == 'Open']
        show_tickets(open_tickets)
        input("Press Enter to go back.")
    elif choice == '2': # See Closed Tickets
        closed_tickets = [ticket for ticket in tickets if ticket.status == 'Closed']
        show_tickets(closed_tickets)
        input("Press Enter to go back.")
    elif choice == '3': # See All Tickets
        show_tickets(tickets)
        input("Press Enter to go back.")
    elif choice == '4': # Respond to Ticket
        while True:
            ticket_input = input("Enter ticket number to respond or 'view' to view open tickets: ")
            if ticket_input.lower() == 'view':
                open_tickets = [ticket for ticket in tickets if ticket.status == 'Open']
                show_tickets(open_tickets)
            elif ticket_input.isdigit():
                ticket_id = int(ticket_input)
                respond_to_ticket(tickets, ticket_id)
                Ticket.save_tickets(tickets, filename)
                input("Press Enter to go back.")
```

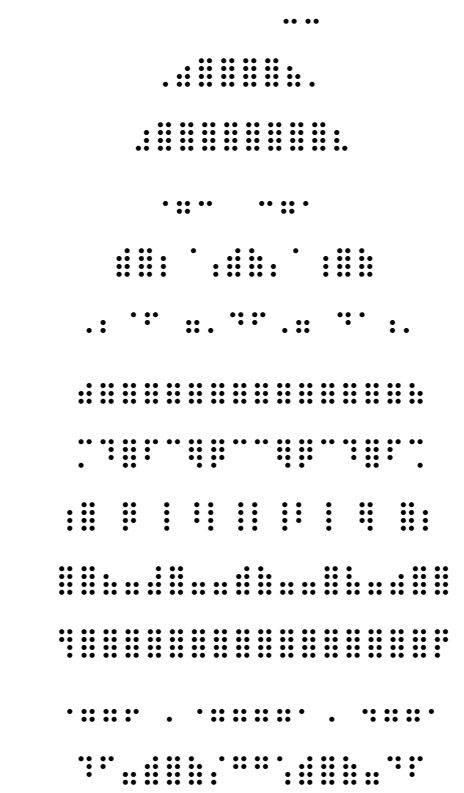
```
            break
    elif choice == '5': # Reopen Ticket
        while True:
            ticket_input = input("Enter ticket number to reopen or 'view' to view closed tickets: ")
            if ticket_input.lower() == 'view':
                closed_tickets = [ticket for ticket in tickets if ticket.status == 'Closed']
                show_tickets(closed_tickets)
            elif ticket_input.isdigit():
                ticket_id = int(ticket_input)
                if not reopen_ticket(tickets, ticket_id):
                    continue
                Ticket.save_tickets(tickets, filename)
                input("Press Enter to go back.")
                break
    elif choice == '6': # Create Ticket
        staff_name = input("Enter Your Name: ")
        staff_id = input("Enter Your StaffID: ")
        while True:
            email = input("Enter Your Business Email: ")
            if "@" in email:
                break
            else:
                print("Invalid email address. Please include @ in the email address.")

        description = input("Please enter a description of the issue or question you have: ")

        if "password change" in description.lower(): # Generate Password
            new_password = staff_id[:2] + staff_name[:3]
            response = f"Your New Password Is: {new_password}"
            status = "Closed"
        else:
```

```
            status = "Open"
            response = ""


        ticket_id = len(tickets) + 2001  # Increment ticket number
        new_ticket = Ticket(ticket_id, staff_name, staff_id, email, description, response,
status)
        tickets.append(new_ticket)
        print("Ticket submitted successfully.")
        Ticket.save_tickets(tickets, filename)
        input("Press Enter to continue...")
    elif choice == '7':
        os.system('cls' if os.name == 'nt' else 'clear')  # Clear the screen before exiting
        print("Exiting program.")
        input("Press Enter to continue...")  # Wait for Enter key before exiting
        break
    elif choice == 'easteregg'.lower():
        print("""
```



```
                    : ⠒ ⣿ ⣿ ⣿ ⣿ ⣿ ⣿ ⣿ ⣿ ⠒ :
                        . ⠤ ⠤ ⠤ ⠤ ⠤ .

""")
        input("Press Enter to continue...")  # Wait for Enter key before exiting




if __name__ == "__main__":
    main()
```

## Ticket Data

- I have created code that sets up a spreadsheet file with pre-defined data as per the clients request, this data in the database.csv file will read any/all tickets created, or altered within the system.

## Ticket Class

- The ticket class is defined with code to allow ticket details to be shown, as well as allowing tickets to be loaded and saved to/from the CSV file.

## Ticket Interactions

- I have created functions to allow tickets to be interacted with. Functions such as ‘respond_to_ticket’, ‘reopen_ticket’, ‘close_ticket’, ‘TicketStats’, ‘show_tickets’, and show_main_menu’ to allow user to interact with the system.

## Main Function

- The main function runs the main menu, allowing users to perform actions such as viewing tickets, responding to tickets, creating tickets, as well as closing/opening tickets.

## Easter Egg

- No reason, I just thought it would be fun to add an easter egg to the prototype as a 'signature' of sorts for crediting the work, which would not exist within the final product once it had been developed and is unlikely to ever be seen by users. Additionally, if the software were to be finished, sent through but not paid for, there would be evidence that I had created the system.

## Debugging / Testing

Throughout the testing/debugging I received many issues, I won't be mentioning the basic ones such as forgetting to add ':' or adding an additional bracket, but I will be going over some of the issues I had faced and how I fixed them.

### Issue with Ticket Submission

- **Problem:** I encountered an issue where users were unable to create a ticket without an error occurring. After investigating, I found that the code was not writing all of the required information as I had not declared all of the variables, such as the contact email, resulting in the system crashing.
- **Solution:** To fix this, I looked at the code surrounding creating tickets, and found instead of:
    - **new_ticket = Ticket(ticket_id, staff_name, staff_id, email, description, response, status)**
    - **It was – new_ticket=Ticket(ticket_id, staff_name, staff_id, description, response, status)** which did not have the email variable in it.
    - By updating the code, I ensured that it was also retaining the email, thus also being the correct structure for the spreadsheet allowing users to create tickets successfully.

### Incorrect Ticket Status

- Problem: I noticed that tickets were being assigned incorrect statuses. As an example, a ticket that had 'password change' inside of it was being responded to with a new password, but was remaining open.
- Solution: To fix this, I made sure that after the new password was shown, which showed itself both on the screen, but also as a ticket response, it should set the variable 'status' to 'Closed', and if it didn't find 'password change' inside of the ticket, it would set the status to 'Open'.

## Error Handling

- Problem: As with most coding projects, in testing you do a lot of unexpected inputs to see what happens and how the system handles it, and as most of those projects, this one is no different – Inputting letters instead of numbers for example, would result in the system crashing.
- Solution: To address this issue, I added code to address error handling. For example, if when creating a ticket you don't have an @ symbol in the email (e.g. randomnamegmailhaha) it prompts the user that it is an invalid email address. Another example, is in the case of reopening a ticket, if the input isn't a valid ID, or 'view' to view the tickets, it will just prompt the user to enter a ticket number. After the prototype is approved, this would be changed to prompt the user as to why it hasn't gone through, as opposed to just prompting the user to do it again.

## Conclusion

With everything said and done, the prototype is now complete and fully functional. The information sent through to the client includes the following:

- Software_Project.py (Python File)
- ReadMe.txt (Includes instructions to display the project)
- IT5016_Assessment_2_20240825 (This file!)

Please read the ReadMe.txt.