REPORT

Table of Contents

[Introduction 3](#_Toc130482641)

[Software Development Life Cycle 4](#_Toc130482642)

[Conclusion 11](#_Toc130482643)

[References 12](#_Toc130482644)

# Introduction

Effective customer service and problem solving are essential in today's hectic company climate to sustain productivity and ensure smooth operations. When it comes to managing and resolving the non-technical and technical difficulties that workers have at work, a ticketing system for the assistance desk is essential. This article provides a working prototype of an within-company customer service ticketing system for the help desk.

The goal of prototype of the planned Help Desk ticketing system is to speed up the procedure for staff members to submit, track, and provide resolution requests. For employees to provide tickets for pertinent information, such as Staff\_id, staff name, staff email, and staff description, the system focuses on giving them an intuitive user interface. Each filed ticket is instantly given a distinct ticket number to facilitate tracking and referencing.

Password update requests are one of the several support request kinds that the design is intended to handle. When a request for a password alteration is made, the system uses the staff member's data to create a new key phrase according to a specified criteria. The IT division may reply to tickets that have been filed, offering input and ending them or opening them again as necessary.

The issuance of tickets also provides crucial data, such as the quantity of opened, closed, and submitted tickets. IT and management division should effectively track the effectiveness of the assistance procedure, they need this information.

The prototype, which is written in Python, is based on two classes—Ticket and Helpdesk—that include the essential features of the apparatus. The Ticket class is used in the Main method of the main class to create and manipulate ticket objects as well as show ticket data and statistics.

Depending on the needs and preferences of the company, a more complete solution may be built on top of this test version of the help desk ticketing system and incorporated into a variety of apps or platforms.

# Software Development Life Cycle

A software project is developed according to a defined method called SDLC, which contains numerous stages and directs the process from inception to conclusion. The steps of SDLC for the prototype source code of the Help Desk ticketing system are listed below:

1. Requirements Gathering:

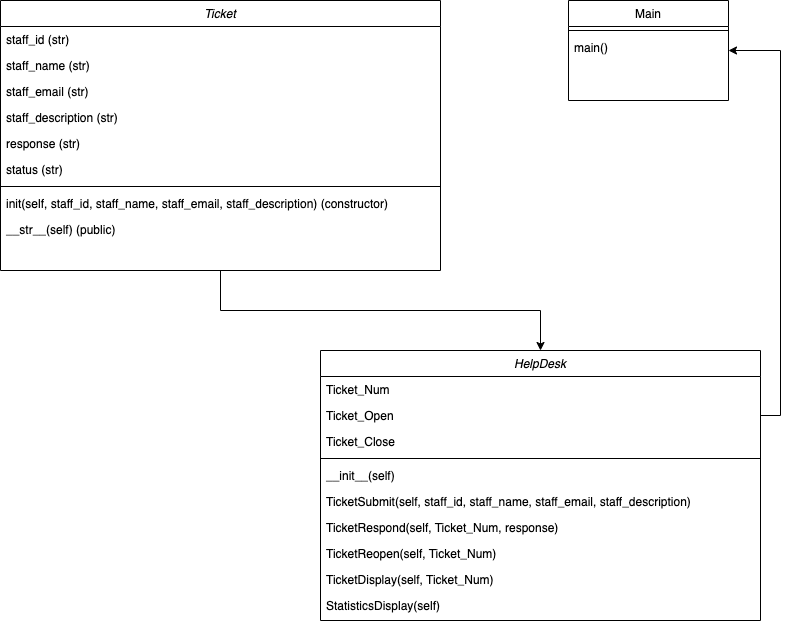
This stage entails collecting information and specifications provided by stakeholders, comprehending the project's desired outcomes, identifying the software's scope and limitations. This lays the groundwork for later phases and aids in developing a clear picture of the undertaking needs. This phase focuses on figuring out if the idea is financially and technically feasible. To make sure that the undertaking can finished within the given limitations, it requires reviewing the resources that are already available as well as time and financial restrictions.

The demands of the customer were collected and examined. The client stipulated that only internal customer tickets should be handled by the system, along with detailed specifications to submit a ticket, answers, statistics monitoring, and updates on ticket progress.

1. Design:

Based on the specifications set in the earlier phases, The system design and software architecture are developed in this stage. The development of a high-level system architecture, outlining components of software and their connections, and defining algorithms, data structures, and component interfaces are all examples of this.

“Ticket” is a term used to describe the attributes of objectss. In the Program class, the Main method was introduced to show how the system works.



1. Implementation:

The actual coding of the programme is done at this stage using the design guidelines. Developers adhere to the coding standards and norms set during the design phase as they produce the code using the programming languages and tools of their choice.

Based on the design, the Main class, the Helpdesk class, and the Ticket class function in the Main class were implemented in the source code, which was developed in Python. The Helpdesk class now has new methods and attributes including PrintTicketInfo, PrintTicketStats, Reopen, ProvideResponse, GenerateNewPassword, and resolve.

**Source Code:**

*#This is a Ticket class with attributes including staff id, staff name, staff email, staff description, response, and status.*class Ticket:  
 *# Initializes the class with staff\_id, staff\_name, staff\_email, and staff\_description parameters.* def \_\_init\_\_(self, staff\_id, staff\_name, staff\_email, staff\_description):  
 *# Set the instance variables equal to the values passed as parameters.* self.staff\_id = staff\_id  
 self.staff\_name = staff\_name  
 self.staff\_email = staff\_email  
 self.staff\_description = staff\_description  
 *# Set response to "not yet supplied" and status to "Open" for new tickets* self.response = "not yet supplied"  
 self.status = "Open"  
 *# Returns a string representation of the Ticket object.* def \_\_str\_\_(self):  
 *# Format the string with the Ticket Number, Ticket Creator, Staff ID, Email Address, Description, Response, and Ticket Status.* return f"Ticket Number: {self.Ticket\_Num}\n Ticket Creator: {self.staff\_name}\n Staff ID: {self.staff\_id}\n Email Address: {self.staff\_email}\n Description: {self.staff\_description}\n Response: {self.response}\n Ticket Status: {self.status}\n"

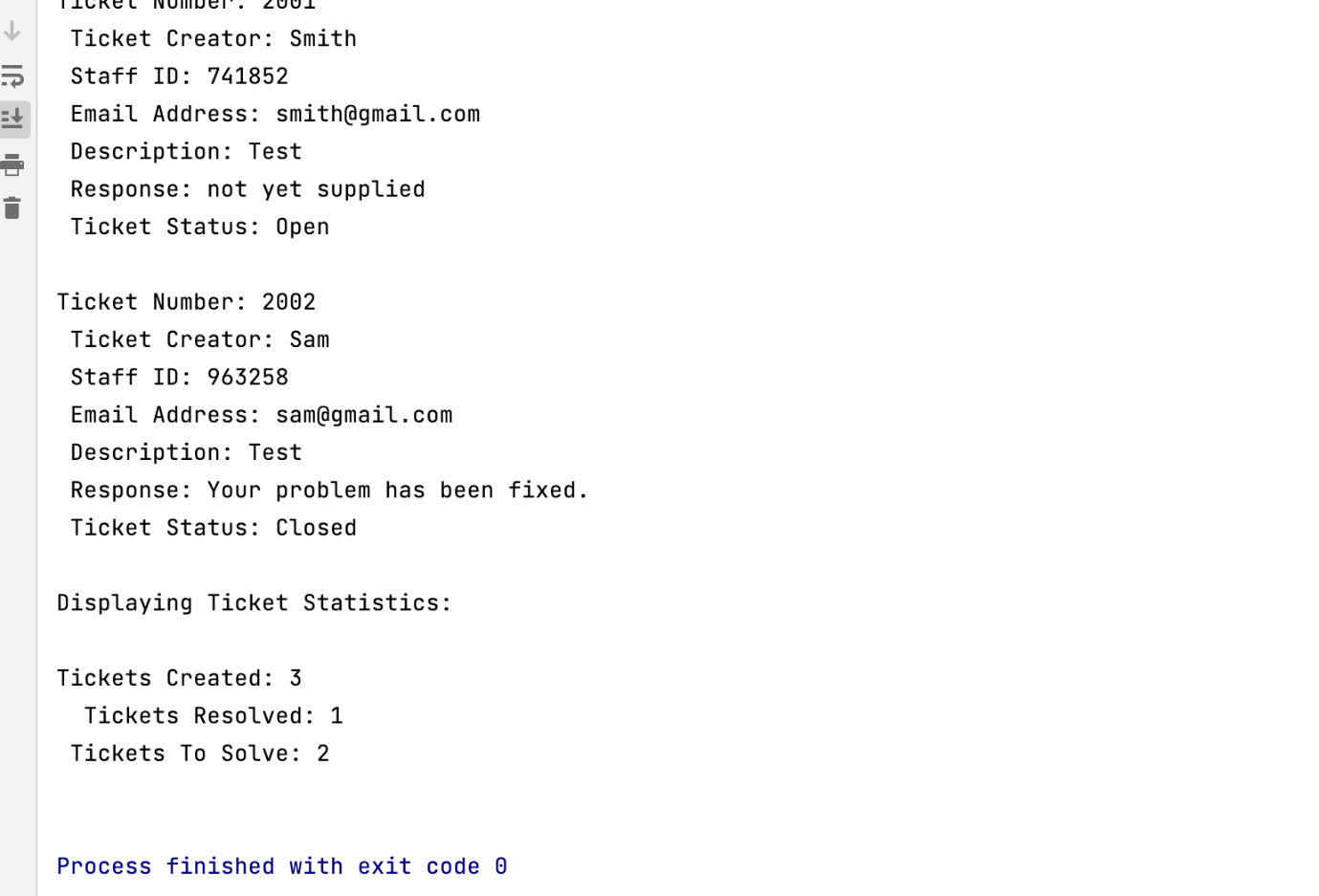
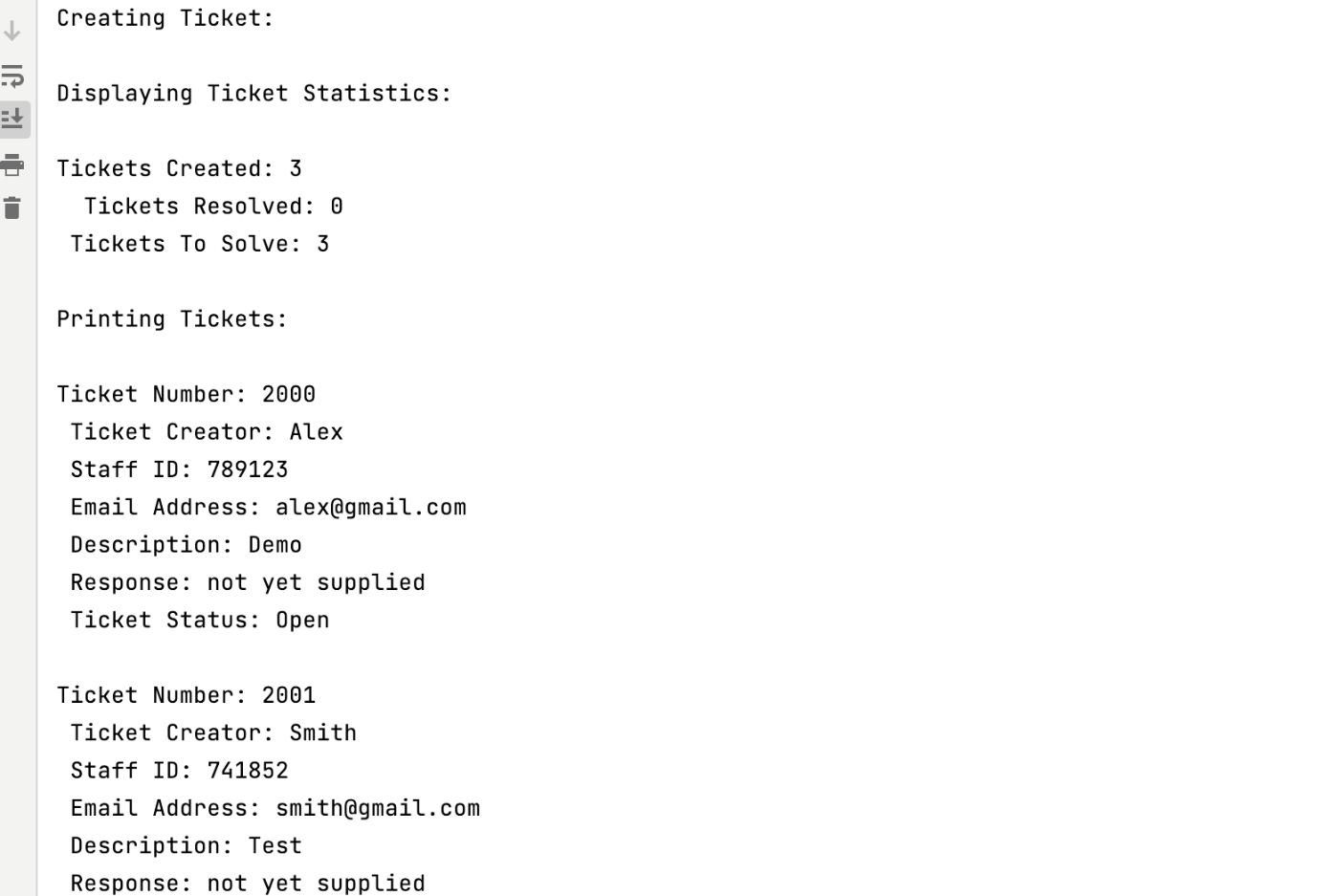
*#Importing the Ticket class from the Ticket module*from Ticket import \*  
*#Creating a HelpDesk class*class HelpDesk:  
 *# Defining class-level variables for ticket numbers, open tickets, and closed tickets* Ticket\_Num = 2000  
 Ticket\_Open = 0  
 Ticket\_Close = 0  
 *# Defining the constructor method for HelpDesk* def \_\_init\_\_(self):  
 *# Creating an empty list to store tickets* self.tickets = []  
 *# Defining a method to submit a new ticket with the given parameters* def TicketSubmit(self, staff\_id, staff\_name, staff\_email, staff\_description):  
 *# Creating a new Ticket object with the given parameters* New\_Ticket = Ticket(staff\_id, staff\_name, staff\_email, staff\_description)  
 *# Adding the new ticket to the list of tickets* self.tickets.append(New\_Ticket)  
 *# Setting the ticket number for the new ticket to the class-level ticket number* New\_Ticket.Ticket\_Num = HelpDesk.Ticket\_Num  
 *# Incrementing the class-level ticket number for the next ticket* HelpDesk.Ticket\_Num += 1  
 *# Incrementing the class-level count of open tickets* HelpDesk.Ticket\_Open += 1  
 *# Checking if the ticket description contains "Password Change"* if "Password Change" in staff\_description:  
 *# Set the response message for the new ticket with a new key phrase using the first 2 characters of staff\_id and the first 3 characters of staff\_name* New\_Ticket.response = f"New Password: {staff\_id[:2]}{staff\_name[:3]}"  
 *# Increment the ticket close count and decrement the ticket open count for the HelpDesk* HelpDesk.Ticket\_Close += 1  
 HelpDesk.Ticket\_Open -= 1  
 *# Set the status of the new ticket to "Closed"* New\_Ticket.status = "Closed"  
 *# Return the new ticket object* return New\_Ticket  
  
 def TicketRespond(self, Ticket\_Num, response):  
 for ticket in self.tickets:  
 if ticket.Ticket\_Num == Ticket\_Num: *# Check if the ticket number matches the given one* ticket.response = response *# Set the response for the matching ticket* HelpDesk.Ticket\_Close += 1 *# Increment the number of closed tickets* HelpDesk.Ticket\_Open -= 1 *# Decrement the number of open tickets* ticket.status = "Closed" *# Update the status of the matching ticket to "Closed"* def TicketReopen(self, Ticket\_Num): *# Define a method called TicketReopen that takes in an instance of the HelpDesk class and a Ticket\_Num as inputs* for ticket in self.tickets: *# Iterate through the tickets attribute of the HelpDesk object* if ticket.Ticket\_Num == Ticket\_Num: *# If the Ticket\_Num of the current ticket is equal to the input Ticket\_Num* ticket.status = "Reopened" *# Update the status of the ticket to "Reopened"* HelpDesk.Ticket\_Open += 1 *# Increment the class attribute Ticket\_Open by 1* HelpDesk.Ticket\_Close -= 1 *# Decrement the class attribute Ticket\_Close by 1* def TicketDisplay(self, Ticket\_Num):  
 *# loop through all the tickets in the list* for ticket in self.tickets:  
 *# if the ticket number matches the provided Ticket\_Num parameter* if ticket.Ticket\_Num == Ticket\_Num:  
 *# print the ticket information* print(ticket)  
  
 def StatisticsDisplay(self):  
 *# print the ticket statistics, using class variables for ticket counts* print(f"Tickets Created: {HelpDesk.Ticket\_Num - 2000}\n Tickets Resolved: {HelpDesk.Ticket\_Close}\n Tickets To Solve: {HelpDesk.Ticket\_Open}\n")

*#Importing HelpDesk module*from HelpDesk import \*  
*#Defining main function*def main():  
 *# Creating an instance of HelpDesk class* helpDesk = HelpDesk()  
 print("Creating Ticket:\n")  
 *# Submitting three tickets with different details* helpDesk.TicketSubmit("789123", "Alex", "alex@gmail.com", "Demo")  
 helpDesk.TicketSubmit("741852", "Smith", "smith@gmail.com", "Test")  
 helpDesk.TicketSubmit("963258", "Sam", "sam@gmail.com", "Test")  
 *# Displaying ticket statistics* print("Displaying Ticket Statistics:\n")  
 helpDesk.StatisticsDisplay()  
 *# Responding to a ticket with ticket ID 2002* helpDesk.TicketRespond(2002, "Your problem has been fixed.")  
 print("Printing Tickets:\n")  
 *# Displaying a specific ticket with ticket ID 2000* helpDesk.TicketDisplay(2000)  
 *# Displaying statistics for tickets with ticket ID 2001 and 2002* helpDesk.TicketDisplay(2001)  
 helpDesk.TicketDisplay(2002)  
 print("Displaying Ticket Statistics:\n")  
 *# Displaying ticket statistics again* helpDesk.StatisticsDisplay()  
*#Checking if the current module is being run as the main program*if \_\_name\_\_ == '\_\_main\_\_':  
 *# Calling main function* main()

1. Testing:

The testing phase entails validating and confirming that the generated software complies with specifications and is error-free. Testing involves a variety of testing procedures including unit testing, system testing, integration testing, and user testing acceptability testing to make that the programme is stable, works well in a variety of scenarios, and is functionally accurate.

To illustrate how the Ticket class works, example cases of tickets were generated, closed, and then recreated in the Main method. Printouts of ticket data statistics were also made to guarantee that the procedures that had been built would behave properly. To verify the system's resilience, more thorough testing with a variety of cases and edges of tests situations is required.



1. Deployment:

The production environment is where the programme is deployed once it has been examined and authorised. Establishing the requisite infrastructure, setting up programme suit the intended setting, and carrying out any necessary data transfer may all be part of this process.

The offered code snippet may serve as the basis for a bigger project. Depending on the needs of the customer, You’d have to incorporate the system into a bigger programme, a console programme, for example. online desktop application, or application.

1. Maintenance:

The maintenance phase starts once the programme has been installed. This covers tasks like bug fixes, feature implementations, and software updates to adapt to alterations to the operational system or requirements. During this phase, it is made sure that the programme is always current, secure, and usable.

The system will need ongoing maintenance to handle security concerns, bug fixes, and the addition of new features depending on user input and changing requirements.

# Conclusion

As a result, the prototype help desk ticketing system given in this article offers a strong framework for the creation of a more thorough and reliable solution for support management. The prototype’s primary goals expediting the procedure for submitting tickets to manage within-company customer service requests inside a business, enabling IT teams to monitor, address, and provide resolution problems quickly.

The prototype was developed in Python and is based on an individual Particular ticket class contains the essential efficiency, making it simple to connect with other platforms and apps. The Ticket class is used in the Main method of the Program class to explain how to generate and modify ticket instances as well as how to display ticket data and statistics.

Although the design is a good place to start, it is crucial to carry out further improvements, thorough testing, and feature expansion depending on the unique demands and specifications of the company. “As a result, we guarantee that the total product is managed. In the end, a competent Help Desk ticketing system helps the business succeed by encouraging productivity, improving communication, and guaranteeing high levels of worker satisfaction.

# References

1. Alton, L., 2016. The Importance of Help Desk Support in a Business. [online] Forbes. Available at: <https://www.forbes.com/sites/larryalton/2016/07/21/the-importance-of-help-desk-support-in-a-business/> [Accessed 23 March 2023].
2. Rockart, J.F., 2003. The Changing Role of the Information Systems Executive: A Critical Success Factors Perspective. In: Galliers, R.D., and Leidner, D.E., eds., 2003. Strategic Information Management: Challenges and Strategies in Managing Information Systems. 3rd ed. Oxford: Butterworth-Heinemann, pp. 31-62.
3. Ivankova, N.V., 2015. Implementing a Help Desk System in a Higher Education Setting: A Case Study of Organizational Change. Journal of Organizational Change Management, 28(4), pp. 658-680.
4. Sommerville, I., 2015. Software Engineering. 10th ed. Harlow, England: Pearson.
5. Pal, S., 2017. IT Service Desk Management: A Complete Guide. International Journal of Advanced Research in Computer Science, 8(7), pp. 2249-9555.