



TICKETQ

Specification & Design

Jeff Shirley

SP20: SYSTEM IMPLEMENTATION: 32534

Table of Contents

Section 1. Customer Problem Statement	3
Section 2. Glossary of Terms	4
Section 3. System Requirements	4
3.1 Functional Requirements	4
3.2 Nonfunctional Requirements	5
3.3 User Interface Requirements	5
Section 4 Functional Requirement Specification	6
4.1 Stakeholders, Actors, and Goals	6
4.1.1 Stakeholders	6
4.1.2 Actors	7
4.1.3 Goals	7
4.2 Use Cases	7
4.3 Use Case Diagram	8
4.4 Traceability Matrix	8
4.5 Fully Dressed Description	9
4.6 System Sequence Diagram	9
Section 5 Effort Estimation	10
Section 6 Domain Analysis	10
6.1 Domain Model	10
6.2 System Operation Contract	11
Section 7 Interaction Diagrams	12
Section 8 Class Diagram and Interface Specification	15
8.1 Class Diagram	15
8.2 Data Types and Operation Signatures	15
8.3 Traceability Matrix	16
Section 9 System Architecture and System Design	16
9.1 Architecture Styles	16

9.2 Identifying Subsystems	17
9.3 Persistent Data Storage	17
9.4 Global Flow Control	17
9.5 Hardware Requirements	17
Section 10 Data Structures	18
Section 11 User Interface Design and Implementation	18
Section 12 Design of Tests	19
Section 13 History of Work, Current Status, and Future Work	20
Section 14 References	20

Section 1. Customer Problem Statement

Our customer is looking for a system that will allow them to track something, in this case we will focus on customer complaints. They want to track these to make their day to day workflow more efficient. The customer has an issue with losing track of important information. Our solution is TicketQ.

TicketQ will allow employees to create a ticket and assign that ticket value. From there they can follow the ticket through the system until it has reached a conclusion. At its conclusion, the ticket will be marked as finished and stored in a database. We will store the completed tickets to be able to call on them in the future for training purposes or as proof of completion.

An example of using our system to track a customer complaint would start with an employee receiving a phone call from a customer to issue a complaint or even receiving a complaint from a customer in front of them at a store.

The employee would receive the complaint and create a new ticket. Then the body of the complaint will be entered along with other identifying information. From there the ticket will dump into the database and will be able to be maintained.

When completed an employee will resolve the complaint according to process put in place by management. If the ticket can be resolved the employee will change the status of the ticket to closed. If the ticket cannot be resolved at this point the ticket will remain open and appear in the live display.

Keeping closed tickets stored on a database will be essential. This will ensure that there is proof of complaints and how they were resolved. This will allow us to take further disciplinary action if needed or even use past examples as training materials for new employees.

Section 2. Glossary of Terms

Glossary of Terms

Ticket – Item that is measurable in the system

User – Employee who created and maintains tickets

Open – Status of ticket that denotes it is a live ticket

Closed – Status of ticket that denotes it is not live and will not show in general display

Live Display – List of tickets that are set as open

Track – The process of following a ticket through from creation to completion.

Ticket Number – Unique number assigned to a ticket.

Database – Digital space where completed tickets are stored.

Live Tickets – Tickets that are currently live in a workqueue and being managed.

Section 3. System Requirements

3.1 Functional Requirements

Number	Priority Weight	Description
REQ-1 Creation	10	Employee MUST be able to create a ticket
REQ-2 Ticket Number	6	System gives each ticket a unique number
REQ-3 Display	8	Display of all ticket information
REQ-4 Search	1	Ability to search the system for specific tickets
REQ-5 Database	9	Storage of ticket
REQ-6 Edit Ticket	7	Employee needs to be able to edit the tickets

REQ-7 Sign-On	3	Employees need to be able to sign-on to the system
---------------	---	--

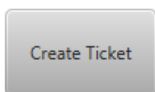
3.2 Nonfunctional Requirements

Requirement	Priority Weight	Description
Functionality	4	Must be able to store data in be able to present information
Usability	10	Accessible by every employee
Reliability	8	Data gathered must be reliable
Performance	6	Must be able to quantify data in a usable fashion
Supportability	2	Must be testable through quality assurance

3.3 User Interface Requirements

Number	Priority Weight	Description
REQ-8 Create Ticket Button	10	Create ticket button is one of the most important functions of the system
REQ-9 Ticket Number	7	The ticket number must be prominently displayed
REQ-10 Display	8	Display all information about the ticket
REQ-11 Search	3	The search field will allow the search and display of open and closed tickets

REQ-8 Create Ticket Button



REQ-9 Ticket number

Ticket Number :

REQ-10 Display

Ticket Display

Ticket Number :	<input type="text"/>	Date Created :	<input type="text"/>
Created By :	<input type="text"/>	Department :	<input type="text"/>
Notes :	<input type="text"/>		
Status :	<input type="text"/>		

REQ-11 Search

Search Ticket

Ticket Number :

Section 4 Functional Requirement Specification

4.1 Stakeholders, Actors, and Goals

4.1.1 Stakeholders

The stakeholders of this project are the employees of the customer company. They will be the one using it and they will be the most invested in TicketQ's success.

4.1.2 Actors

The actors of the system will be people interacting with it daily: employees, managers and Human Resources.

4.1.3 Goals

The goals of the users are to create and maintain tickets. They will also want to be able to track trends and get ahead of any personnel or stock issues that arise.

4.2 Use Cases

Use Case 1 (REQ-1 Creation)

The user will receive a complaint and create a ticket in TicketQ.

Use Case 2 (REQ-2 Ticket Number)

When an employee or manager creates a ticket, TicketQ will create a unique ticket number.

Use Case 3 (REQ-3 Display)

The user will be able to display all information assigned to a specific ticket.

Use Case 4 (REQ-4 Search)

Users will search for tickets to gather data for tracking purposes.

Use Case 5 (REQ-5 Edit)

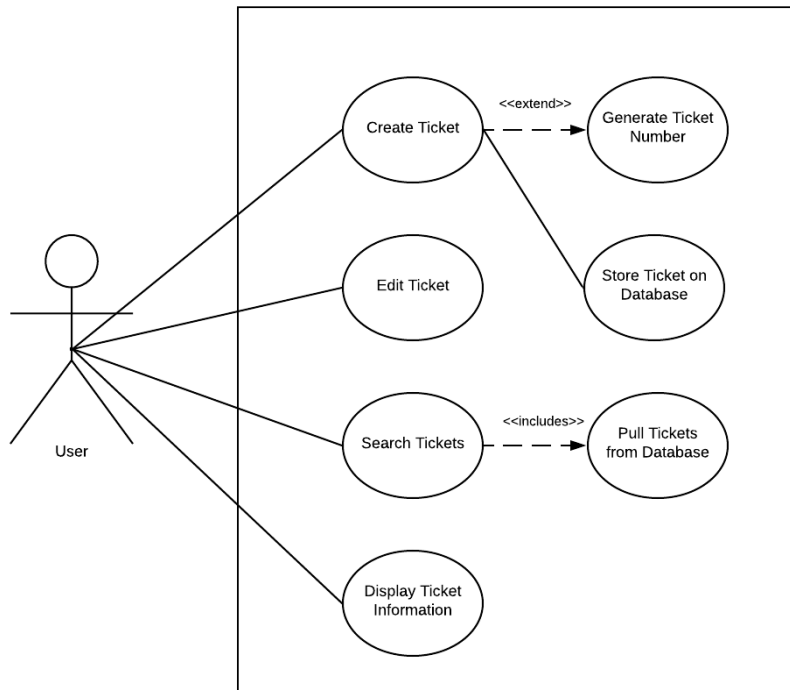
Users will edit ticket information as needed, mainly notes and status.

Use Case 6 (REQ-6 Sign-On)

Users will use unique usernames and passwords to sign-on. This will allow the ticket to grab this information for the “Created by” field.

4.3 Use Case Diagram

TicketQ Use Case Diagram



4.4 Traceability Matrix

	REQ-1	REQ-2	REQ-3	REQ-4	REQ-5	REQ-6	REQ-7	REQ-8	REQ-9	REQ-10	REQ-11
UC 1	X	X						X			
UC 2		X						X	X		
UC 3			X		X					X	X
UC 4				X	X						X
UC 5				X	X					X	

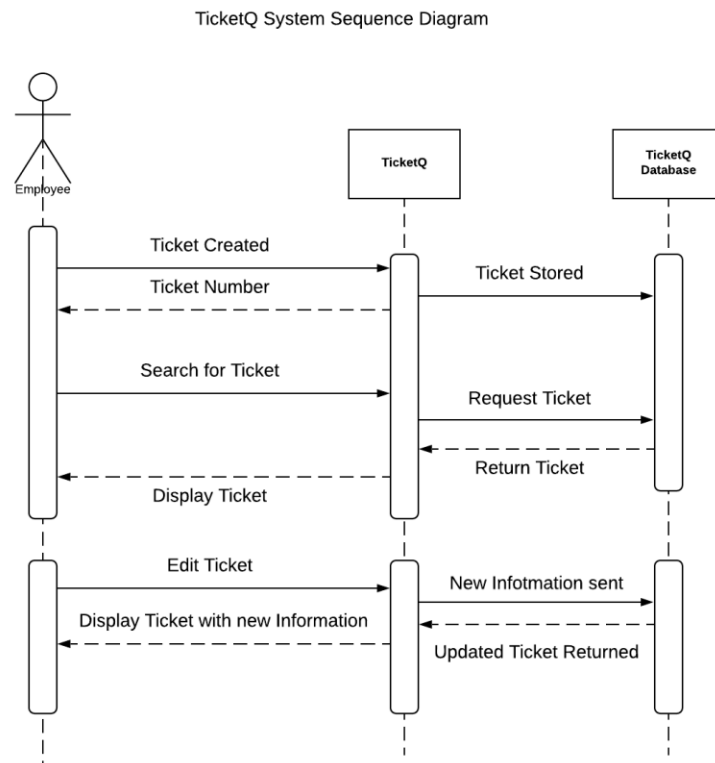
UC 6							X				
------	--	--	--	--	--	--	---	--	--	--	--

4.5 Fully dressed Description

When an employee creates a ticket, they will start with the complaint. They will receive the complaint and enter the information into TicketQ. This information will include department, status, and complaint. Information like ticket number, date created, and who created the ticket will be auto populated by the system.

Once the ticket is created it will dump into the database and become searchable. Once an employee finds the ticket, they will be able to edit and maintain that ticket.

4.6 System Sequence Diagram



Section 5. Effort Estimation

Use Case 1 – User will click File in menu bar and choose Create Ticket. The user will enter the Department, Status, and Notes. Finally, the user will click the create ticket button and be given a ticket number. 5 Clicks and keystrokes depend on the amount of information entered.

Use Case 5 – User will click File in the menu bar and then click Search Ticket. The user will be taken to the Search Ticket screen. The user will then enter the desired ticket number and click Search. This will take the user to the Display Ticket screen. From here the user will click Edit Ticket and be taken to the Edit Ticket Screen. The user may then update the ticket information. 4 clicks and keystrokes depend on the amount of information edited.

Section 6 Domain Analysis

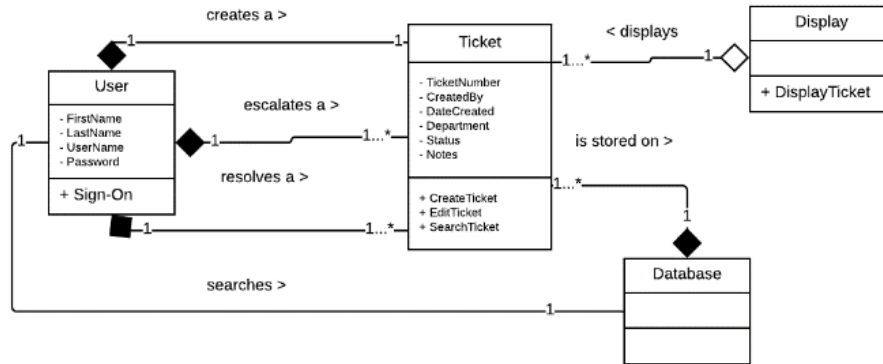
6.1 Domain Model

The TicketQ Domain Model is a display of 4 concepts: User, Ticket, Display, and Database. The User has many connections with the Ticket since the user will be manipulating the ticket multiple times. The User has a composition relationship with the ticket since a ticket would not exist if the user did not create it. The user can also search the database.

The Database has a composition relationship with the Ticket because if the ticket isn't stored on the database than it no longer exists.

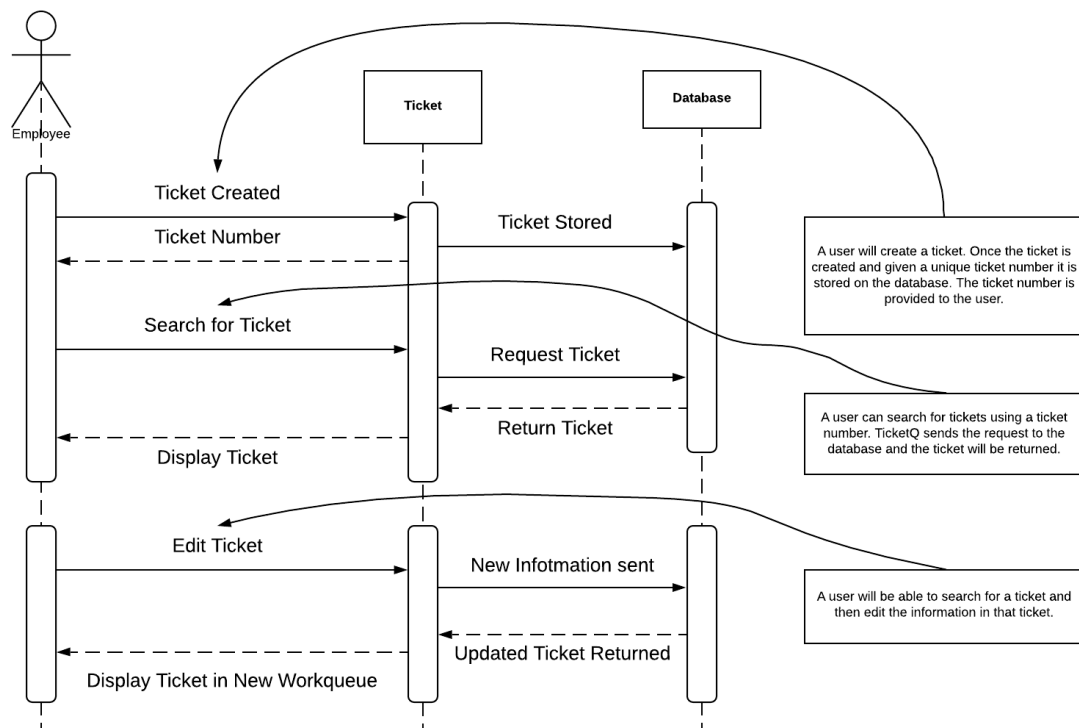
TicketQ Domain Model

Jeff Shirley



6.2 System Operations Contract

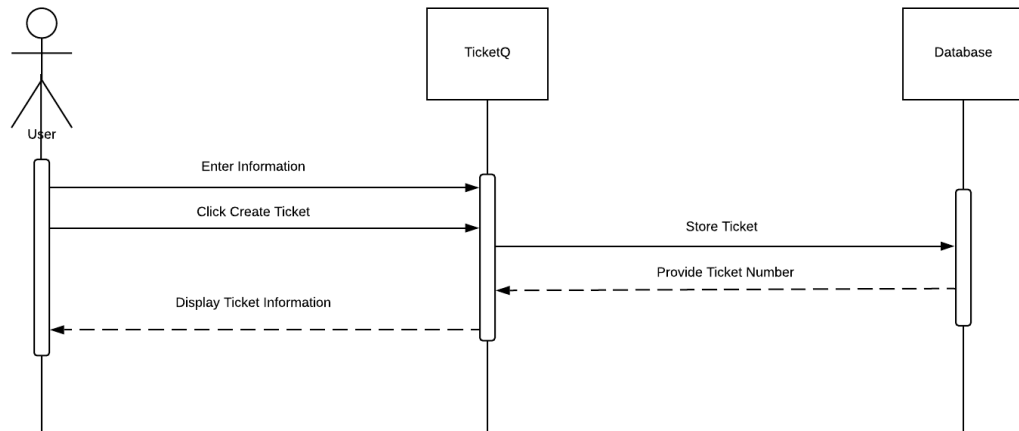
TicketQ System Sequence Diagram



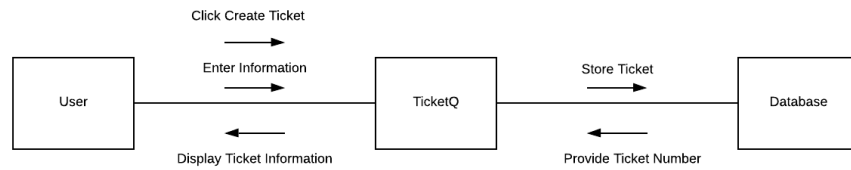
Section 7. Interaction Diagrams

Use Case 1: User receives a complaint and creates a ticket.

Use Case 1 Sequence Diagram
Jeff Shirley

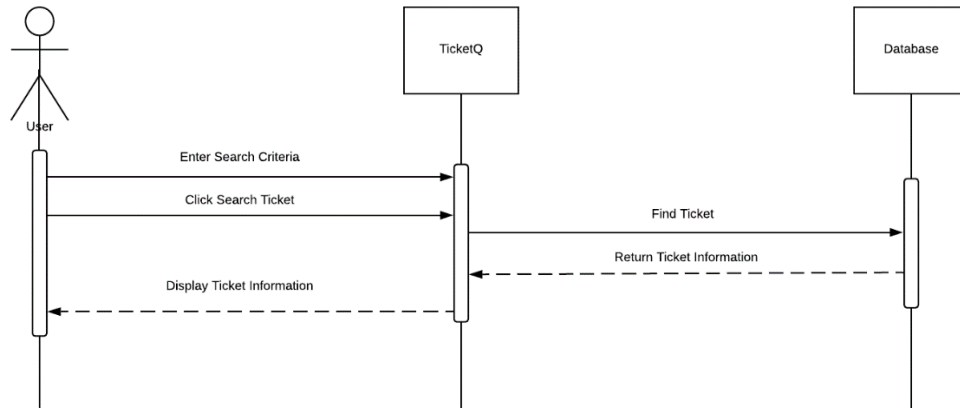


Use Case 1 Collaboration Diagram
Jeff Shirley

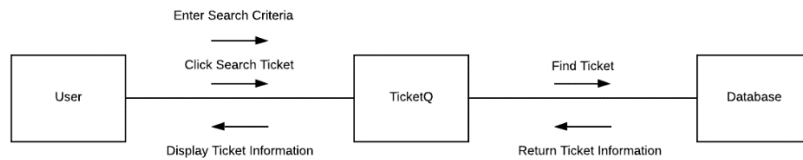


Use Case 2: User searches for a ticket and displays information.

Use Case 2 Sequence Diagram
Jeff Shirley



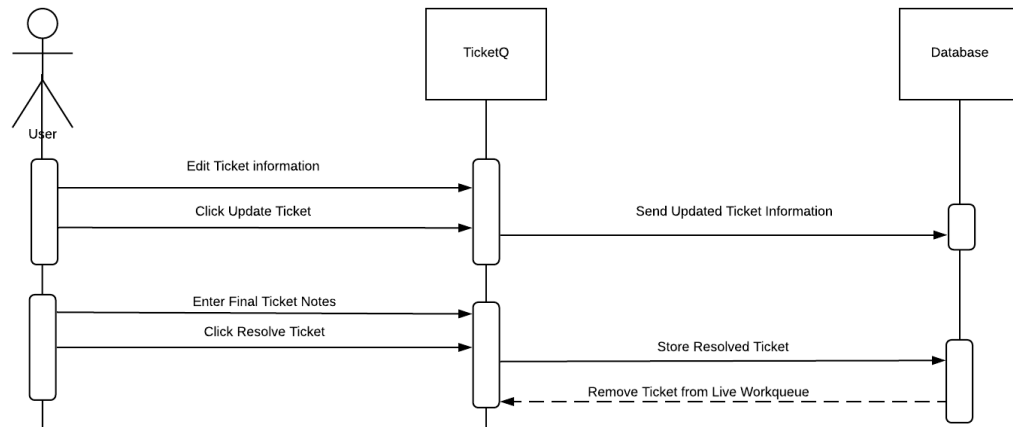
Use Case 2 Collaboration Diagram
Jeff Shirley



Use Case 3: User edits/resolves ticket information.

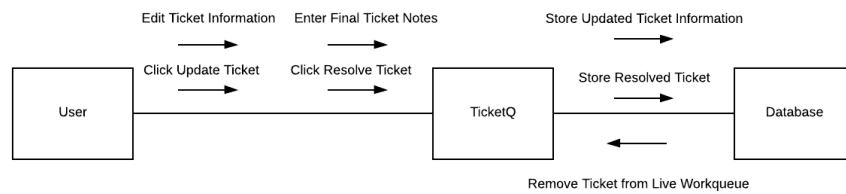
Use Case 3 Sequence Diagram

Jeff Shirley



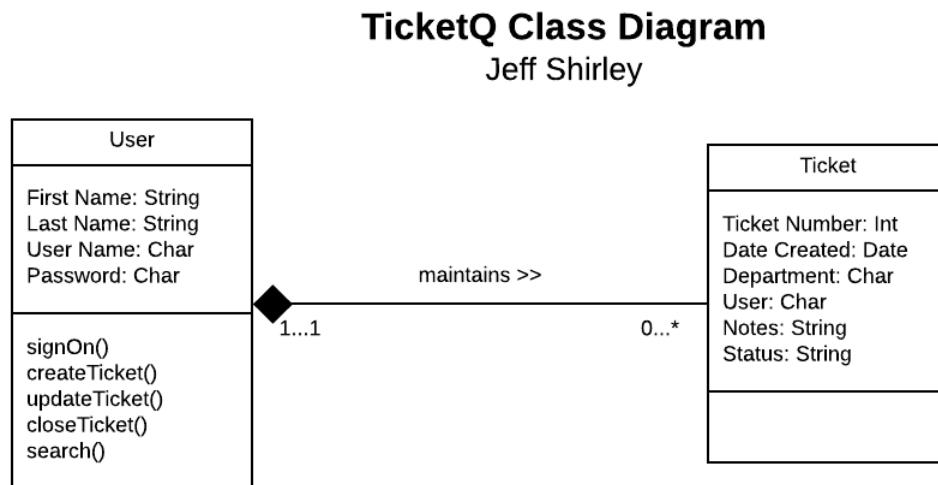
Use Case 3 Collaboration Diagram

Jeff Shirley



Section 8. Class Diagram and Interface Specification

8.1 Class Diagram



8.2 Data Types and Operation Signatures

We have narrowed down to 2 classes. First the User will have the following attributes: First Name, Last Name, User Name, and Password. First and Last Name will be strings consisting of the Users actual name. User Name and Password will be characters due to the User being able to create these themselves so they can contain letters and numbers. The User will be able to preform many operations such as signOn(), createTicket(), updateTicket(), closeTicket(), and search().

Next the Ticket will have the follwing attributes: Ticket Number, Date Created, Department, User, Notes, and Status. Ticket Number will be an integer provide by the system. Date Created will be a Date field in the following format, 00/00/0000. Department will be a string. User will be a character field that copies the User Name from the User who creates the

Ticket. Notes will be a string field accepting large chunks of information. Status will be a character field with either a Y for open or N for closed. The Ticket will not perform any operations itself, it will just hold information and be manipulated by the User.

8.3 Traceability Matrix

Req't	PW	UC1	UC2	UC3
Req1	8	x		
Req2	4	x		
Req3	6		x	x
Req4	1			
Req5	7		x	
Req6	3			x
Req7	8	x		
Req8	5	x		
Req9	6		x	x
Req10	2			
Req11	3		x	x
Total PW		25	22	18

This chart displays the priority weights of the three main Use Cases presented earlier in this document.

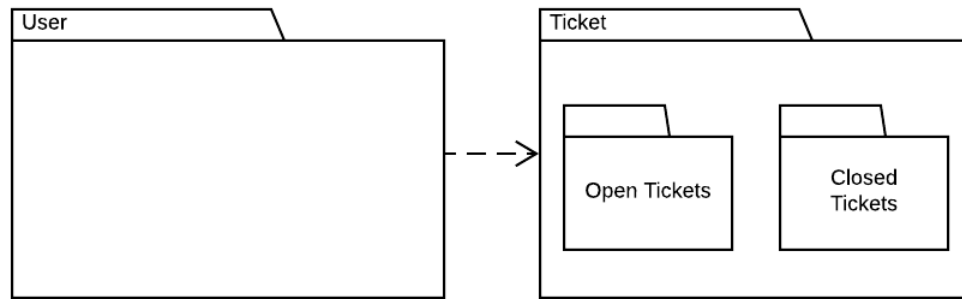
Section 9. System Architecture and System Design

9.1 Architecture Styles

The architecture style of TicketQ is very much an event-driven design. Nothing happens without the user making a decision. The user will prompt the system to create a ticket, search for a ticket, edit a ticket, and close tickets. It will even be users who generate user names and passwords for themselves or others.

9.2 Identifying Subsystems

TicketQ Package Diagram



9.3 Persistent Data Storage

This system will use a relational database to store information. The two main classes, User and Ticket, will store information on the database. User will search Tickets and manipulate the information. The relational database will be created using SQL.

9.4 Global Flow Control

TicketQ is an event-driven system. The user will determine how and when to initiate and manipulate tickets. Nothing will happen in this system without a users input. The user will sign on the system using a self created user name and password. The user will create, search and manipulte tickets. The only thing the system does without user input is store information.

9.5 Hardware Requirements

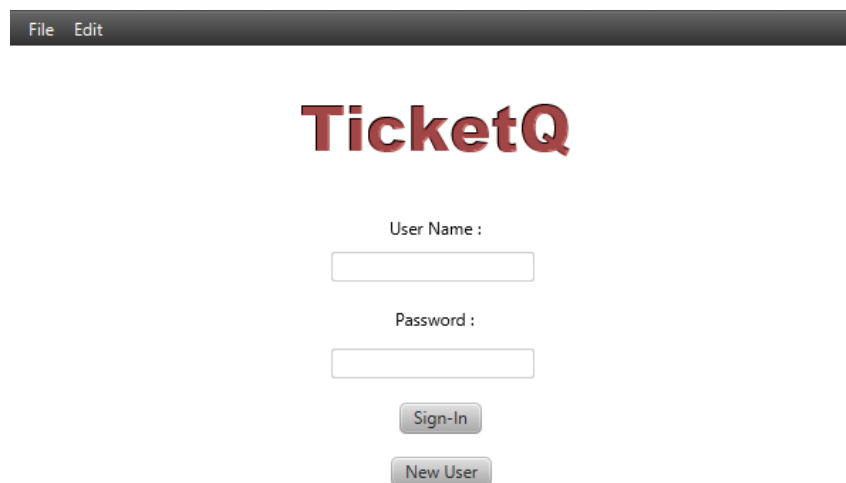
The basic hardware requirements for the system consit of a screen display capable of displaying grayscale, keyboard, and a mouse. The resolution for the basic app will be 800x480

and will not have any color components. Users will use keyboard and mouse to enter data and click buttons. The user will need 1GB of free disk space for storage of the system. No other requirements will be needed for the system.

Section 10. Data Structures

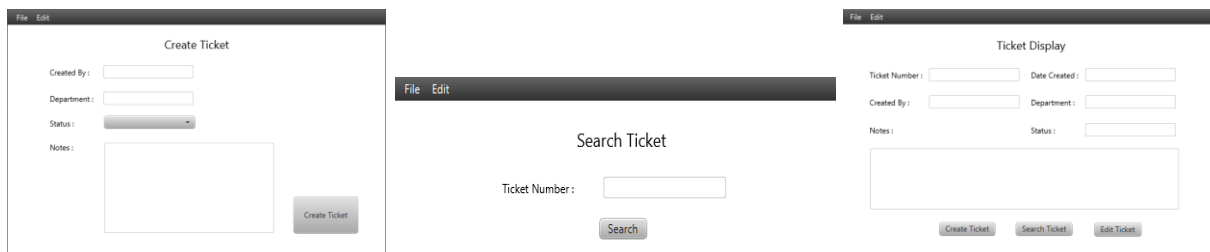
The data structure used in this project is an Array List. The array list allows us to be flexible in the fact that we can keep creating objects, tickets, to add to it. It also allows us to define the tickets and store the full array of information in each ticket.

Section 11. User Interface Design and Implementation



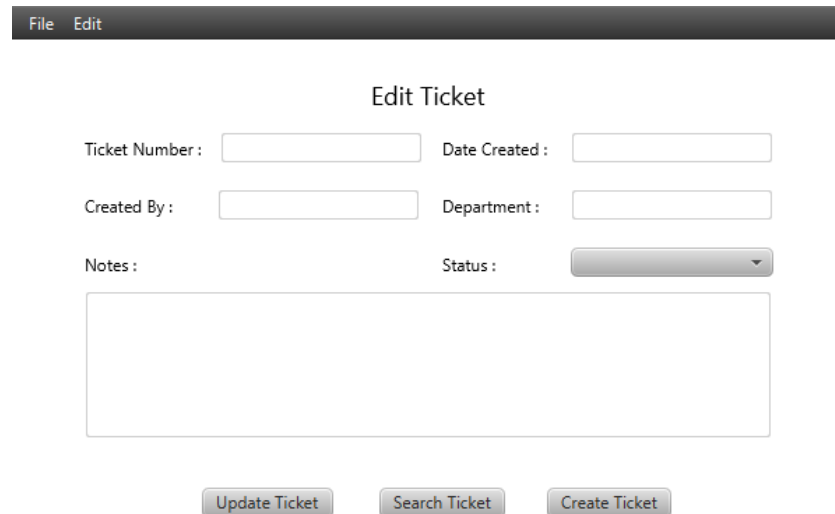
The image shows the login interface of the TicketQ application. At the top, there is a dark grey menu bar with the text "File Edit". Below the menu bar, the title "TicketQ" is displayed in a large, bold, red font. Underneath the title, there are two input fields: "User Name :" and "Password :". Below the password field, there are two buttons: "Sign-In" and "New User".

The TicketQ UI has been updated to provide a more intuitive user experience. The user will be able to preform any actions through the File menu: create, search, and display.



The image shows three windows of the TicketQ application. The first window on the left is titled "Create Ticket" and has a menu bar with "File Edit". It contains input fields for "Created By:", "Department:", "Status:" (a dropdown menu), and "Notes:". There is a "Create Ticket" button at the bottom right. The middle window is titled "Search Ticket" and has a menu bar with "File Edit". It contains a "Ticket Number:" input field and a "Search" button. The third window on the right is titled "Ticket Display" and has a menu bar with "File Edit". It contains input fields for "Ticket Number:", "Date Created:", "Created By:", "Department:", "Notes:", and "Status:". There are three buttons at the bottom: "Create Ticket", "Search Ticket", and "Edit Ticket".

Once a user creates a ticket it will take them to the Ticket Display page. The user will be able to execute many actions from this page. They will be able to immediately create a new ticket, search for other tickets, and even edit the currently displayed ticket.



The screenshot shows a web application interface for editing a ticket. At the top, there is a dark grey header bar with the text 'File Edit'. Below this, the title 'Edit Ticket' is centered. The form contains several input fields: 'Ticket Number :', 'Date Created :', 'Created By :', and 'Department :', each followed by a text input box. Below these, there is a 'Notes :' label followed by a large text area, and a 'Status :' label followed by a dropdown menu. At the bottom of the form, there are three buttons: 'Update Ticket', 'Search Ticket', and 'Create Ticket'.

Section 12. Design of Tests

Our system is fairly easy to test. When creating a ticket the system provides the ticket number and date. This leaves the remaining inputs for the user. Fortunately, these inputs are text fields and can not have invalid entries. So for unit testing we just need to make sure that our main functions provide the automatic data for the tickets and accept user input.

When testing the system we just need to verify that the array list is holding our tickets. From there we can add other functionality like searching and editing tickets. We will also verify that our users can log in. We will also want to ensure that our system can display all open tickets.

Section 13. History of Work, Current Status, and Future Work

The history of work on this project is the current status. I am working diligently to finish the UI and code to have a finished product. The project initially was going to use a database to store and maintain the tickets. Then it switched to an ArrayList. The issue with that was the fact the save state was not reliable. We are back to integrating a database and naturally this has caused some extra work.

In future versions of TicketQ I would bring back the department specific workqueues. These would display all open tickets assigned to them and allow users to grab tickets from there. I would also elaborate on the search function, allowing users to search through parameters other than ticket number.

Section 14. References

<https://www.lucidchart.com/>