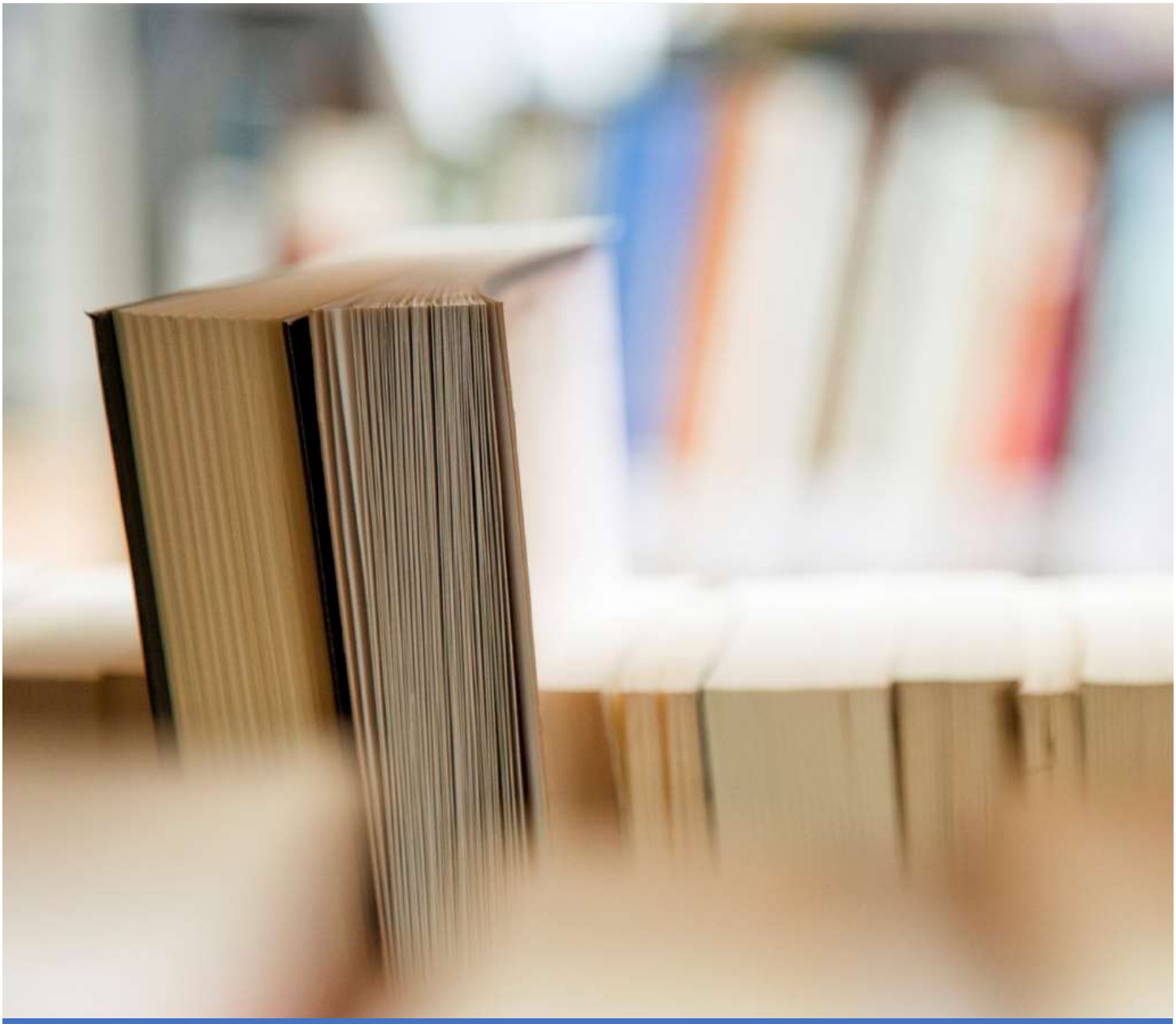


BitTicket

Alex Hughes & Mark Pepere

Contents

Introduction

In most organisations, help desk systems are very important and provide key support to employees as well as the company. The system to be developed will be designed for a hypothetical school. BitTicket will be implemented on Qt and developed in C++.

Purpose and Scope

The purpose of a help desk system is to allow end-user to submit their issues/incidents to a system for assessment and solution. The system allows centralized operation through collaboration to help the end-user solve said issue/incident. This system can be used for assigning end-user queries to the relevant department or agent while tracking outcomes and storing information.

Development Approach

This project will be approached using the waterfall/agile method. The team felt that the combination of the two methods was well suited as it is simple to manage each phase of the project. This is an advantage as the timeframe is only 5 weeks long with precisely defined requirements and allows a good amount of reviewing at the end of each phase. Utilising peer programming techniques and scrum meetings allows the team to hit deadlines and manage workloads.

Waterfall Steps:	Agile Techniques:
• Requirement analysis • Design • Development • Testing	• Peer programming • Daily scrum meetings

Software Requirements analysis and specification.

Identification of Functional and non-Functional requirements.

Parameters	Functional Requirement	Non-Functional Requirements
Requirement	It is mandatory	It is non-mandatory
Capturing type	It is captured in use case	It is captured as a quality attribute
End-result	Product feature	Product properties
Capturing	Easy to capture	Hard to capture
Objective	Helps you verify the functionality of the software	Helps you to verify the performance of the software
Area of focus	Focuses on user requirement	Concentrates on the user`s expectation and experience
Documentation	Describe what the product does	Describes how the product works
Product Info	Product features	Product properties

Jelvix jelvix.com

Requirements elicitations from clients.

Interview with Farhan

1. What are your reasons for looking for a helpdesk Package?

Yoobee college has a growing internal setup that needs Software to sort and categorize the overwhelming demand in our IT (Information Technology) section.

2. How many different divisions do you have within the school?

Yoobee college has multiple divisions which include IT, Staff/Tutors, Students, Health and Safety also offering 3 tiers of IT support.

Tier 1 (Low) Standard low-level PC, Mac, and laptop support.

Tier 2 (Medium) Support for Username and password creations and updates.

Tier 3 (High) New software installs New Software requests.

3. What is the current Schooling size Tutors, Students, IT Division, Health, and safety division?

Yoobee college has around 3000 students and 170 staff members.

4. What is the level of IT support offered by the school now?

Tier 1 mostly but some Tier 2 Auckland with the data center.

5. What does your troubleshooting process look like now?

Contact emails a staff member. Staff member resolves ticket using the software support if needed.

SCCM is used for all Microsoft remote control, patch management, operating system deployment, network protection, and numerous other services.

6. What is the standard operating system used throughout the school?

Windows 10, macOS 11. Windows 11 integration will be within a year.

7. Are there any types of standard Software packages used at the college?

Microsoft Office, Microsoft Office 365, Adobe suite, Autodesk, Animation software, Visual Studio, Visual Studio Code, Figma, and Qt.

8. Could you please list the different standard types of hardware used?

Standard hardware (PCs, Mac, Mouse, Keyboard, etc.), Docking stations, Projectors

All in max machine and printers.

9. Is there a level of access for students vs administration?

Not much is provided currently.

10. what are your most common logged issues?

Software installs and requests typically take around 5 to 6 months, which can be challenging at times. July is the cut-off for software for next year's package rollout.

Ethnographic research.

ZenDesk

(https://www.zendesk.com, n.d)

Created a Test account for ZenDesk to see how current market ticketing/helpdesk systems operate.

Signup/Login was simple, asked for basic information including:

- First and last name
- Email
- Company name
- Password
- Verification of email

Get Started Tab – Main Dashboard

Assessment of the layout: The layout consisted of a side menu that separated the core function of the system. The first core function was the 'Getting started' (see figure 1)

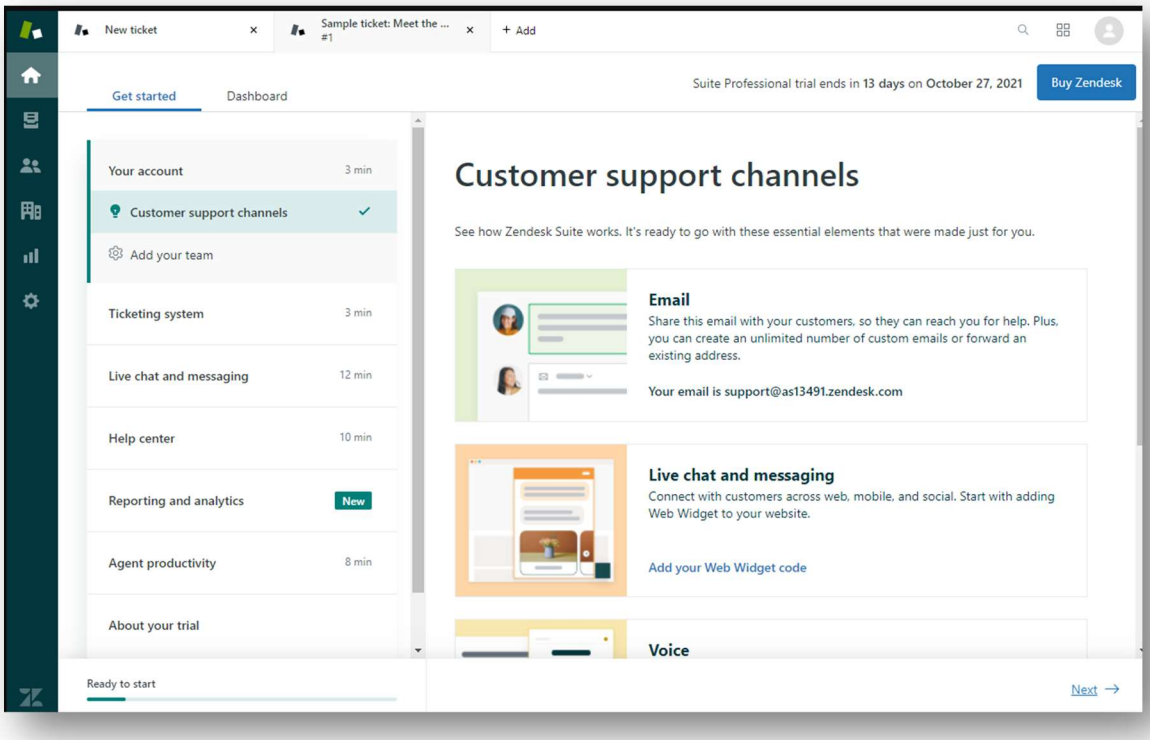


fig 1.

As can be seen above this section is related to the teaching and setting up of the program. From a user's perspective, this is a key aspect as it allows companies to self-teach and become operational quickly. The 'Get Started' Menu includes tutorial videos and key statements on functionality.

View Tab

The second tab from the menu named 'Views' is the core ticketing management system. (See figure 2 & 3)

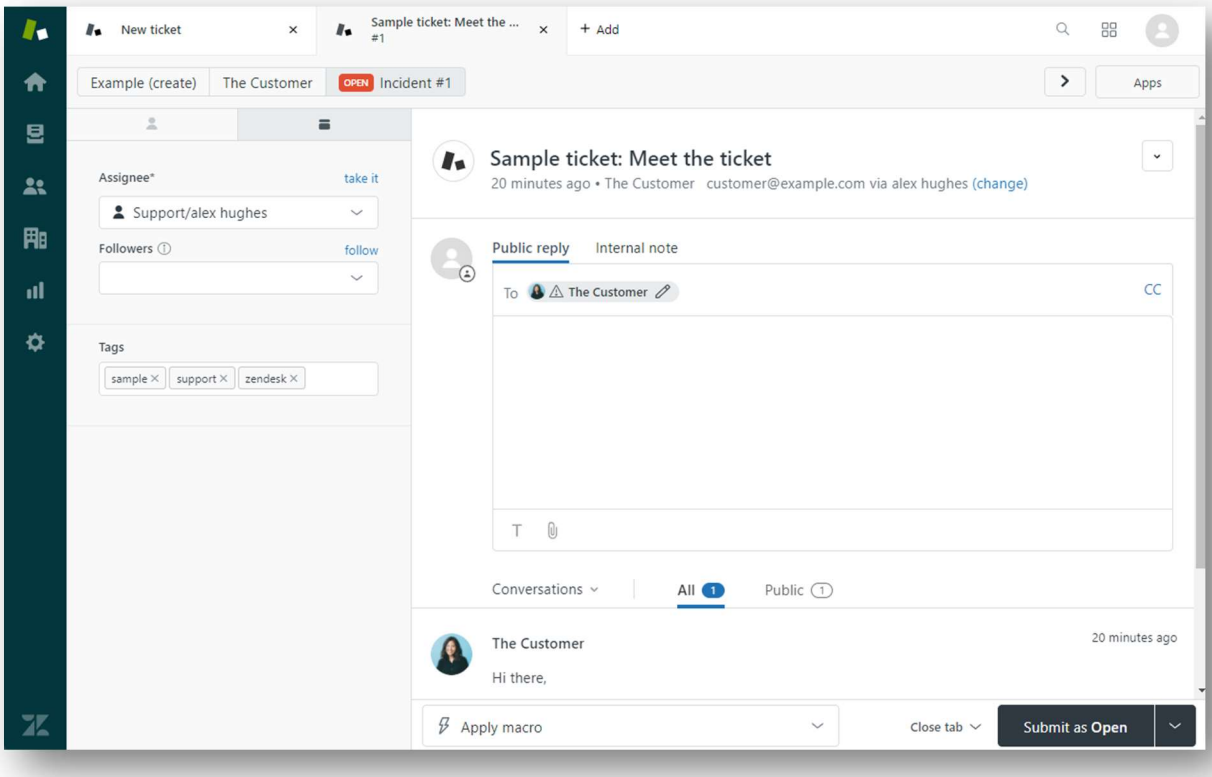


fig2.

The main 'view page' allows the opening and processing of tickets and provides the company with an immediate overview. This allows the company to get a quick broad view of all types of tickets and the operator can quickly assess and prioritize what needs attention.

*Additional thoughts: This feeds the reporting side of the system.

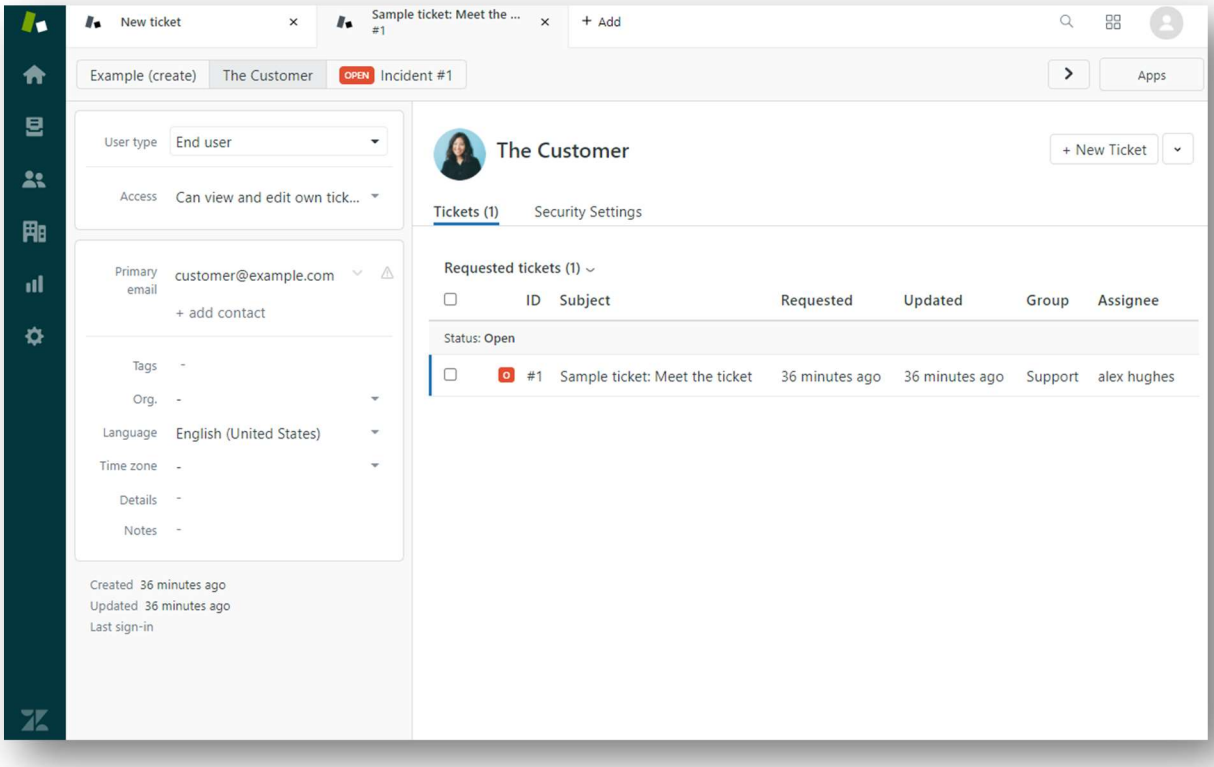


fig 3.

In fig 3 above, the system has provided a sample ticket, this ticket is a basic customer email ticket. bote key features:

- Public reply and internal note
- Assignment tag and followers
- The customer info tab (see figure 4)
- Manual Ticket creation (new Ticket)
- Conversation options
- Submission options (see figure 5)
- Tags
- Secondary tabs (see figure 6)

*Additional thoughts: companies can interact with multiple tickets at once, this is likely an efficiency feature that directly a-lines with real-time customer interaction. The function to have multiple tabs open at the top of the screen in figure 3 allows the company to jump between tickets quickly. simple feature but effective.

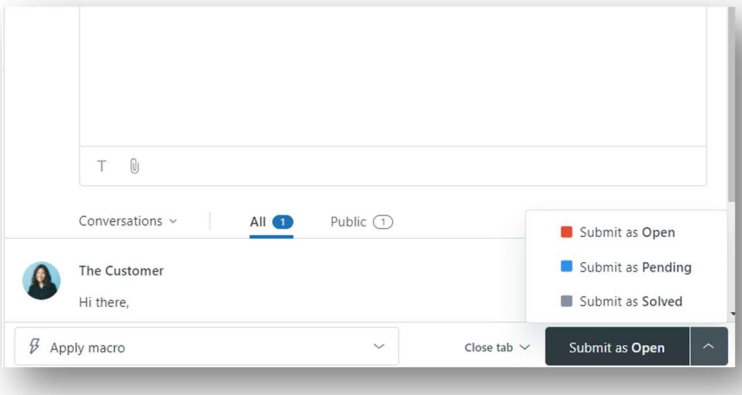


fig 4.

The main customer information tab allows the company to manually update the customer's information as well as permissions. This tab also shows the current and previous tickets lodged but the customer.

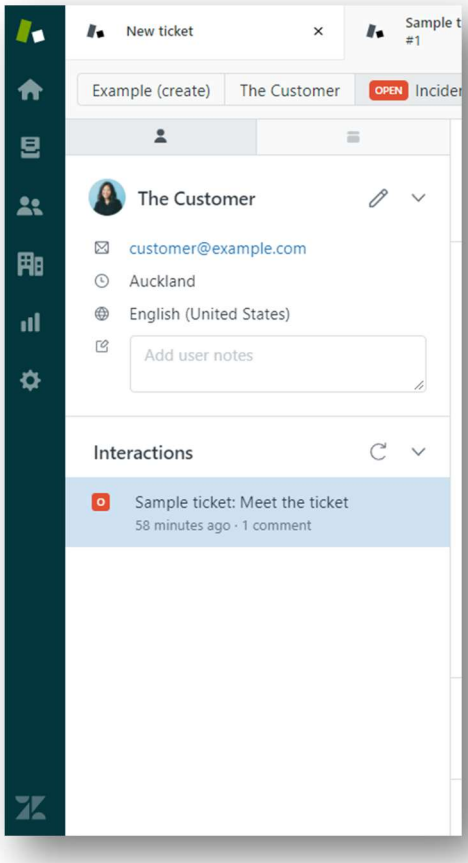


fig 5.

Submission of an incident is clear and easy for the company to assign a category on the push button.

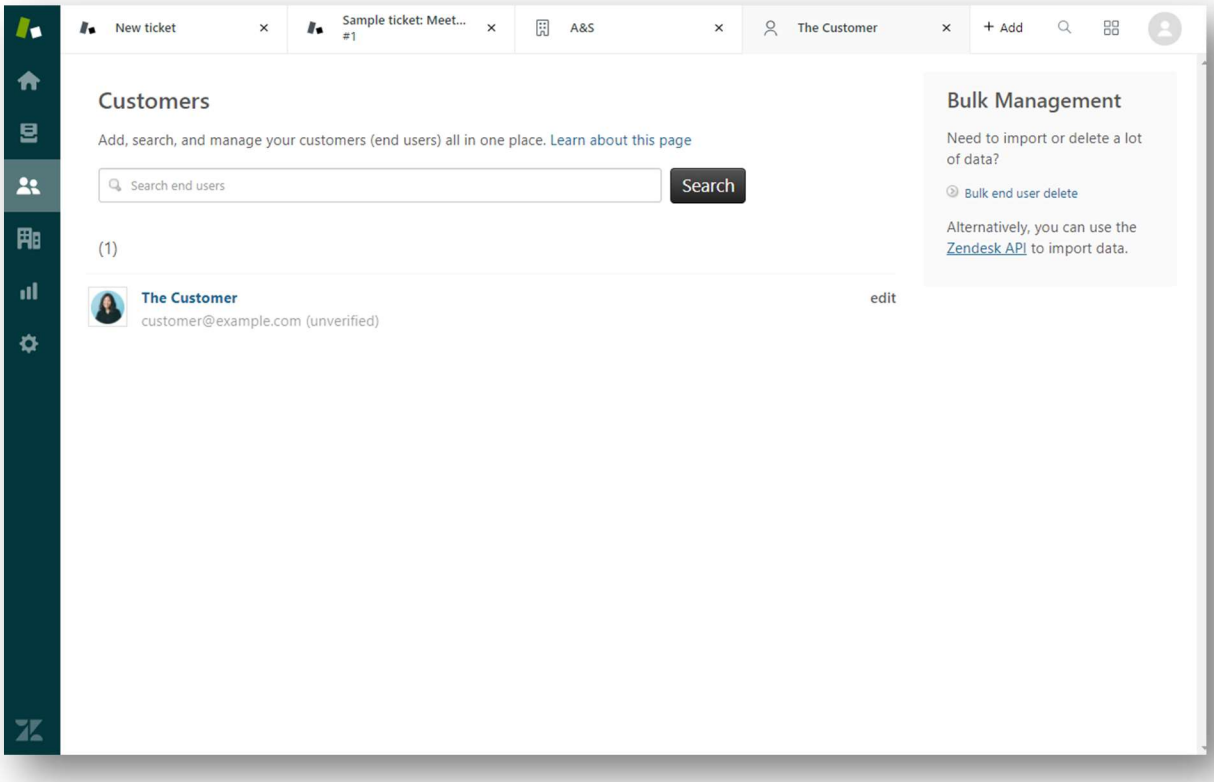


fig 6.

In figure 6 above small features such as a secondary tabular system allows the company to view basic information and any previous interactions.

Customers Tab & Organisation Tab

The 'customer tab' is the function to add, search and manage customers all in one place as figure 7 describes.

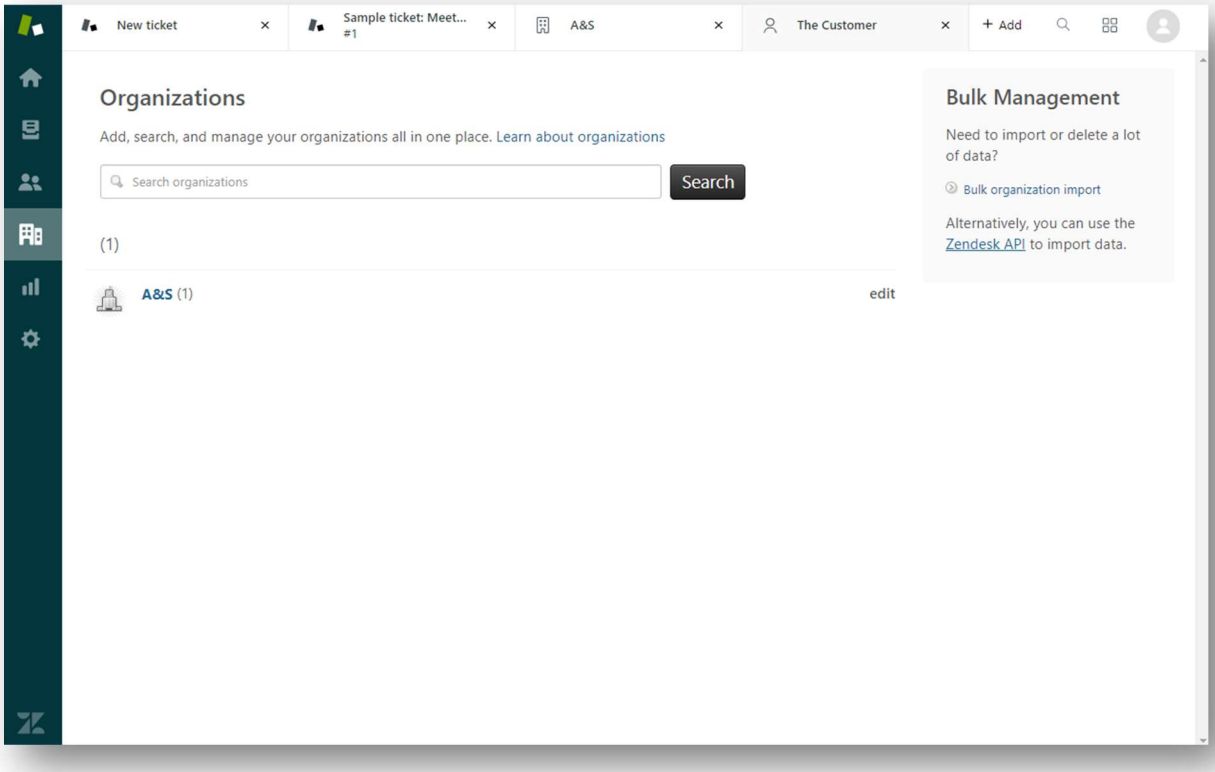


fig 7.

The separation of the feature means that bulk managing a large customer base becomes much easier for the company. If the company wishes to import customer data into the system this accommodates the process, this feature, while simple, is attractive to a company with pre-existing databases and a smart selling point.

The key feature that is offered in this tab is the search function for absolute speed and efficiency.

The flow of the program means that once a customer has been searching on, the company can then edit the information associated, this is linked to the customer information seen in fig 4 under the 'view tab'

The 'organization tab' is a mirror function of the 'customer tab' it also allows add, searches, and manage organizations all in one place. (See figure 8)

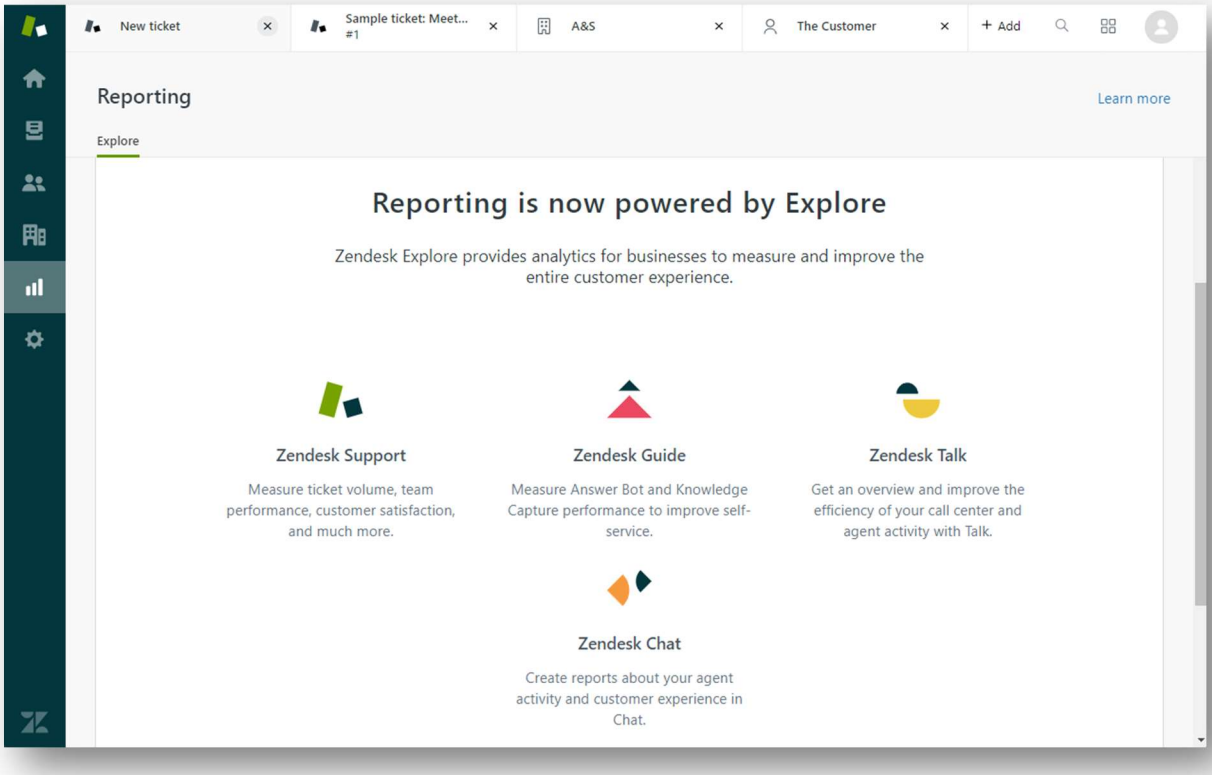


fig 8.

Reporting Tab

As expected, this feature pulls all the information collated from the program into a specific tool that can be used by the company for many reasons. Measuring KPIs (Key performance indicators) would be the most obvious.

Companies benefit greatly from analytics as it allows management to understand primary aspects that may be understaffed, poorly managed and areas for improvement. This sounds very glass half full, but this information shows areas that are exceeding expectations as well as areas that need work.

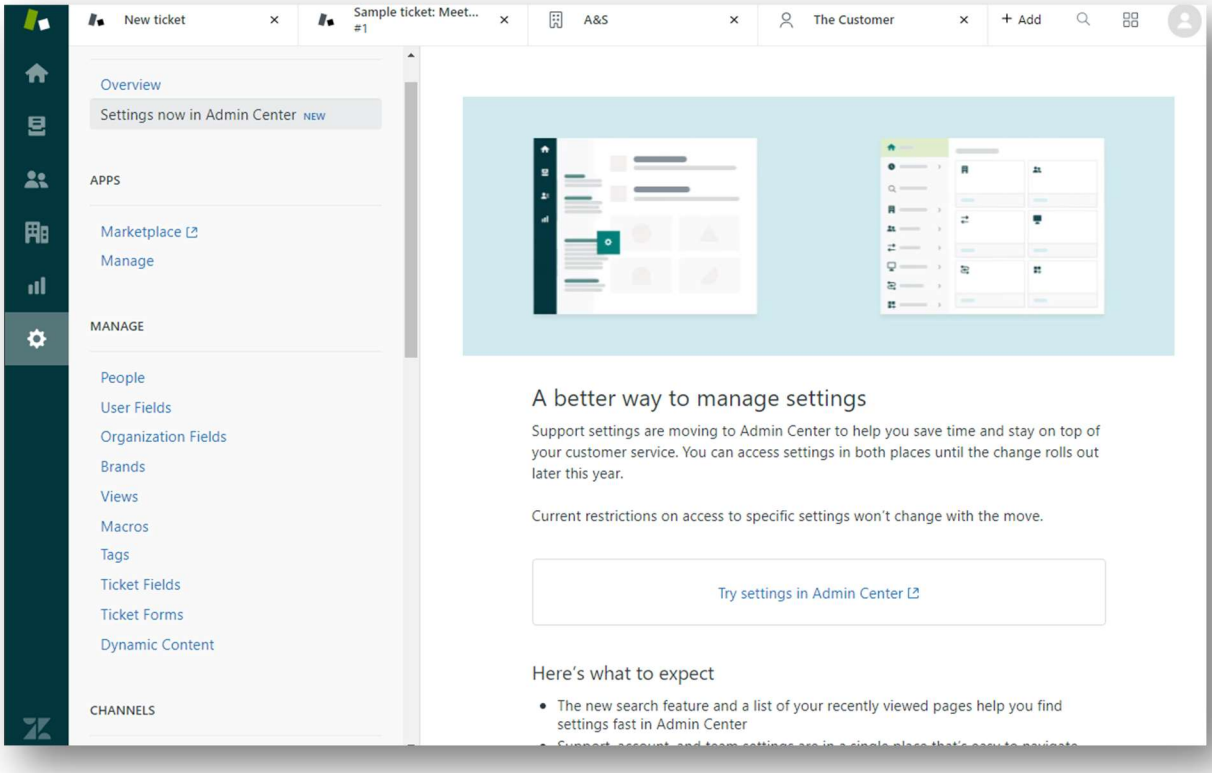


fig 9.

Admin Tab

This is the setting tab of the program, it provides the ability to manage all features that are offered, this allows the company to personalize the program in a variety of ways.

Management of applications, people, organizations, tickets, tags, etc.

It is broken into five subcategories that allow complete support to the company using the program.

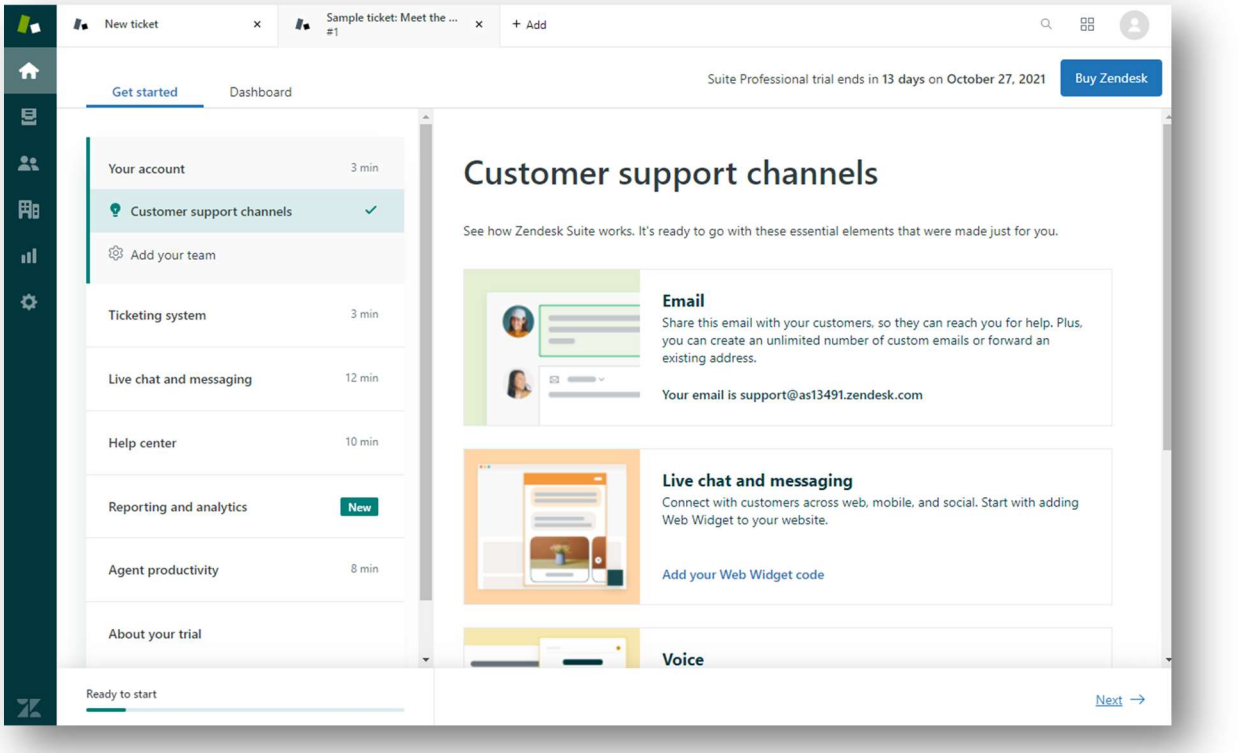


Fig 10.

The basic breakdown of this tab is as follows.

Admin home - Apps, the management of integrations

Manage - Management of customers i.e. People, organizations, brands, etc.

Channels - Management of socials, customer interaction functions, API, and SDK.

Business Rules - Rules that apply to tickets to allow better allocation to relevant people within the business, automation, and service level agreements.

Settings - program settings including security, account, staff, permissions, etc.

While necessary, a large part of this functionality may be out of scope for the ticketing project and time frame given

Conclusion of ZenDesk

In conclusion, ZenDesk seems to be a very well-developed ticketing program with many other facets. The flow of the program is very nice. With limited experience, grasping most of the key features presented was not difficult. The inter-connected tabs allow for easy maneuverability around the program which delivers a sleek feel.

The GUI looks smart and easy to look at. This would be expected as many people will spend hours of company time looking at this program.

SolarWinds Service Desk

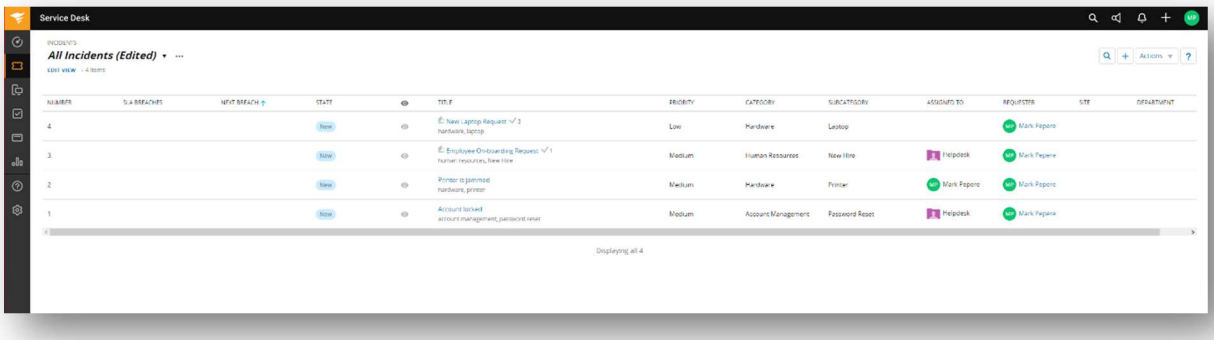
(https://www.solarwinds.com, n.d)

Created a Test account for SolarWinds to see how current market ticketing/helpdesk systems operate.

Signup/Login was simple, asked for basic information including:

- First and last name
- Email
- Company name
- Password
- Verification of email

Main Dashboard – Overview



Summary

Assessment of the layout: The main dashboard is compact with features that include the following:

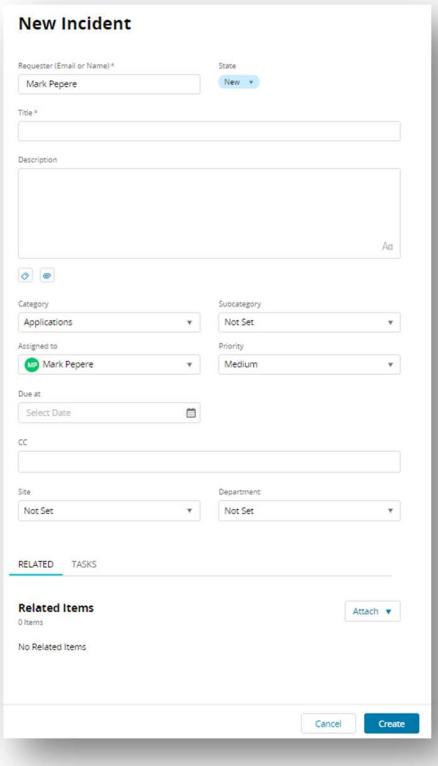
- Creation of ticket.
- Modification of existing ticket.
- Filtering of any category.
- Viewing of analytics.

The dashboard provides visual feedback on the performance of support and relevant data – quick modification of any ticket active or complete. Insight and analysis are also included, the dashboard is a great tool for quickly reviewing key metrics as well as actions that need to be taken.

Key takeaway points

The dashboard here has been well thought out. Quick Overview of any active ticket and I like the fact that the Administrator makes simple and easy modifications to any part of the ticket from the dashboard itself.

Ticket Creation



Requester: The user's Full name who has requested support.

State: The State of the ticket itself. New, Open, or Closed.

Title: Title of the ticket and topic relevance.

Description: A description of the ticket placed by the user. Generally, information can help streamline the process of resolving the ticket.

Category: Ticket category is the overview title i.e., hardware, accounting, human resources, networking, software, etc.

Sub-Category: Sub-category is activated once the category has been chosen for example if the user selects hardware within Category, the sub-category will offer desktop, laptop, peripherals printer.

Assigned to: This allows whoever opens the ticket up to assign it to anyone in support

Priority: The priority dropdown offers the admin the right to set the level of urgency to the ticket.

Due date: Due date for the ticket must be resolved by.

CC: tag in more admin if needed for a higher level of support or focused support from someone very predicant in that area that the ticket falls in.

Site: Current site that the ticket relates to i.e., Auckland campus, Wellington campus, or Christchurch campus.

Department: Department the ticket will fall under facilities, finance, Human resources, or marketing.

Attachment: Admin or user can attach any relevant documentation relating to the ticket.

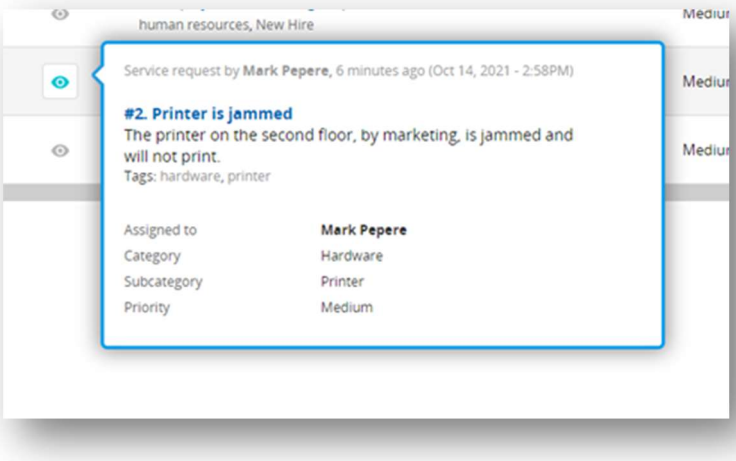
Summary

Ticket creation is a core and fundamental part of the helpdesk ticketing system. SolarWinds have made it easy, while in the filtration process of any given ticket, the administration can add in new fields if required and not listed while in a state of creating any given ticket.

Key takeaway points

SolarWinds have done a great job at refining the creation of a ticket process. A feature that I haven't seen in similar products is the ability to CC other administrators into the ticket allowing for extra input if the ticket requires. I can see this process as very beneficial when a administer comes across a ticket that can sometimes be outside the box.

Ticket Overview



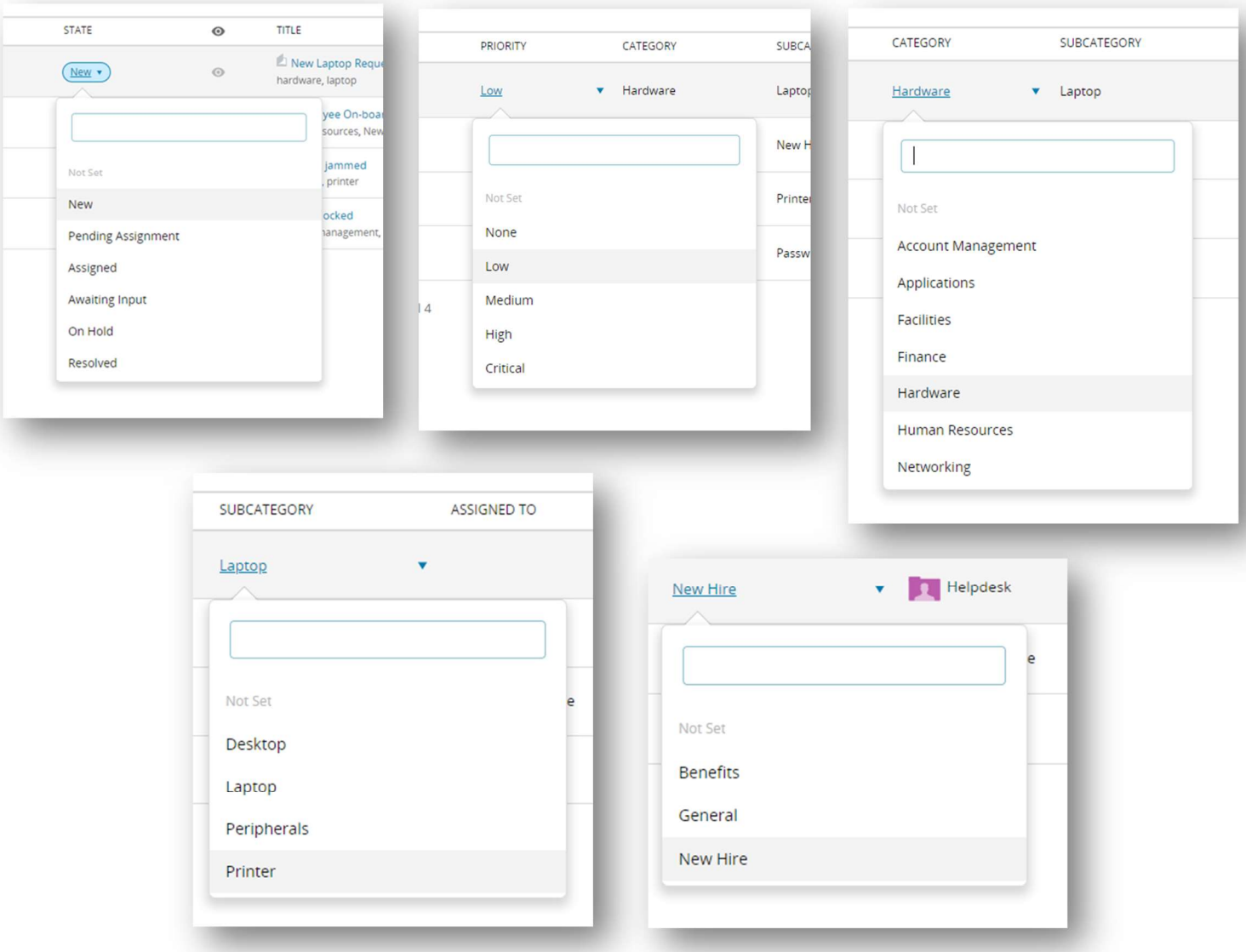
Summary

SolarWinds offers a nice feature for giving Administration a nice and quick way to overview the ticket. No modifications can be made in this state.

Key takeaway points

The is a feature I noticed within Qt to show a tooltip. This could be a feature we consider implementing as the quick view of a ticket can be a very powerful tool.

Modify State - Priority Filter – Category Filter – Subcategory filter



Summary

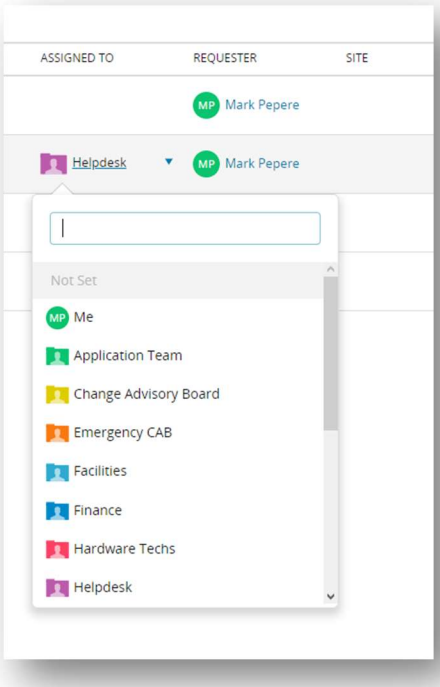
The above if modifying a state, Priority, Category, and subcategory filter of the ticket and is a nice, needed feature that gives the administration the right to update a ticket's state at any time.

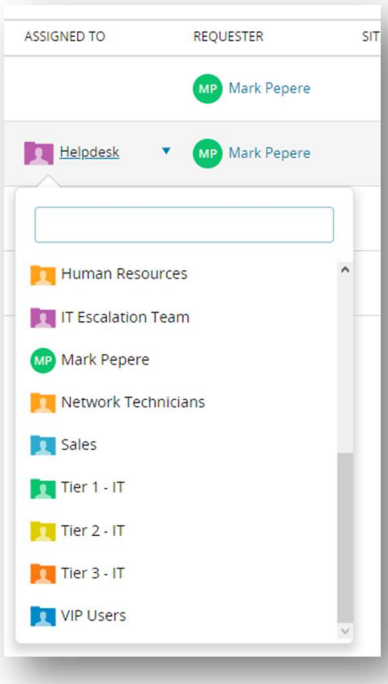
Above is an example from SolarWinds of the different states that the ticket can be updated to. This is the filtration process to ensure the ticket is sent to the right support department.

Key takeaway points

This could be presented within our project as a combo-boxes. Nice and simple.

Assigned To





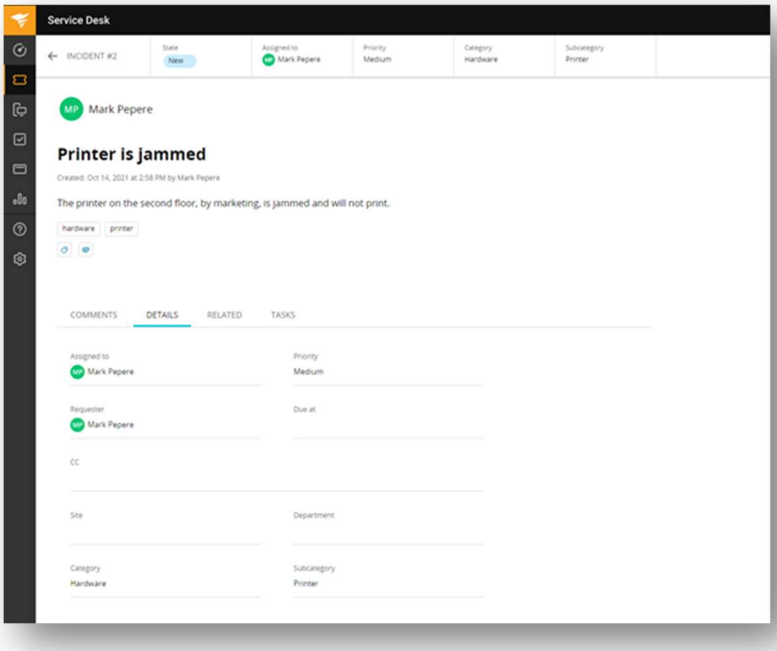
Summary

Like the above examples of the filtration process of the ticket. This feature allows the ticket to be assigned to any given person or department. The tier 1, tier 2, and tier 3 IT support indicates the type of support that is required for the ticket.

Key takeaway points

Having this prepopulated will probably work nicely within our project given the administer quick access to the right person or department.

Incident Card (Ticket Overview in modification state)



Summary

Once a ticket is created the Administrator can go into any given ticket and modify the current state to better reflect the change in the ticket. If the ticket has been actioned like above where the print is jammed and needs attention and then has been fixed, the Administrator will update accordingly.

Key takeaway points

Something to note here there are extra tabs for added comments. If the ticket changes state this with where the administrator will update the comments section. This is important because if another administer opens the ticket, they can formularize themselves with the ticket history by reading the previous comments and the states it has passed through.

Conclusion of SolarWinds

In conclusion, SolarWinds seems to be a very flushed-out ticketing system with options that can suit any business profile. With a nice easy layout, it was very quick to navigate to any part of the program with a small number of clicks. SolarWinds leads itself nicely and creates an easy-to-use environment experience for a new user. The filtration process of a ticket I found was thought out covering every needed topic and if there wasn't a needed categorization you could just simply add one

The GUI look is easy on the eye and looks very smart and simplest at the same time. Tying in great functionality with a good look makes this package an asset to review.

Functional & Non-functional Requirements

Req	Functional requirements	Must/Want
FR001	The administrator will be able to auto allocate a number to a ticket.	Must
FR002	The system will allow the admin to assign the ticket to the department.	Must
FR003	The system will allow the admin to add/edit tags to the ticket.	Must
FR004	The system will allow the admin to determine outcomes.	Must
FR005	The system will allow the admin to set the urgency of the ticket using 3 options (Urgent, Semi Urgent, not urgent).	Must
FR006	The system will allow the admin to set the priority using 3 options (High, mid, low).	Must
FR007	The system will allow the admin to add/edit dates and times to tickets.	Must
FR008	The system allows space for the user to describe issues for the admin to assess.	Must
FR009	The system will allow for the admin to add/edit the level of IT support or corresponding department hierarchy.	Must
FR010	The system allows users to add contact details to tickets during submission.	Must
FR011	The system allows the admin to add contact details to tickets during submission.	Must
FR012	The system will allow categories for channels to record how the ticket was submitted.	Must
FR013	The system will allow the user to add final comments/reviews of performance.	Must
	Non-Functional requirements	
NFR001	The system will only make the admin sign into the program for personal record-keeping and auto-fill information.	Want
NFR002	The system will track multiple stats and KPIs to export data on performance.	Want

User and System Requirements
SRS - Software Requirements Specification

Req ID:	**Snow Card**
FR001	Type: Functional Actor: Administration Requirement: The administrator will be able to auto allocate a number to a ticket Justification: Core Function of system Priority: High Originator: Developer Team Description: As an admin, I want to add a unique ticket number to each ticket that is submitted. Fit Criterion: If an admin can add a unique ticket number to a new ticket. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 19, 2021 (Creation Date)
FR002	Type: Functional Actor: Administration Requirement: The system will allow the admin to assign the ticket to the department/agent Justification: Core Function of system Priority: High Originator: Developer Team Description: As the primary admin assigning the ticket to the right department or agent means that the incident can be dealt with quickly and by the right personnel. Fit Criterion: Assigning the ticket to the correct department or agent, and they receive it. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 19, 2021 (Creation Date)

FR003	Type: Functional Actor: Administration Requirement: The system will allow the admin to add/edit tags to the ticket Justification: The core function of the system Priority: Medium Originator: Developer Team Description: As an admin, I would like to add tags to the ticket to further connect the ticket to other points of interest. Fit Criterion: Tags are visible on the ticket info and relevant to the incident/issue Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 19, 2021 (Creation Date)
FR004	Type: Functional Actor: Administration/Agent Requirement: The system will allow the admin to determine outcomes by changing the status of the ticket and changing the status of the incident. Justification: The core function of the system Priority: High Originator: Developer Team Description: As the admin/agent I should be able to change the status of the ticket to Closed and the status of the incident to Solved, Not Solved, Ongoing) Fit Criterion: Changing the status of the ticket/incident should reflect within the information of the ticket and remove its display from the system. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 20, 2021 (Creation Date)

FR005	Type: Functional Actor: administrator Requirement: The system will allow admin to set the urgency of the ticket using 3 options (Urgent, Semi Urgent, not urgent) Justification: The core function of the system Priority: High Originator: Developer Team Description: As Admin I need to assign urgency to a ticket based on its merits Fir Criterion: adding urgency to a ticket means that the department/agent will know how quickly a solution needs to be. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 19, 2021 (Creation Date)
FR006	Type: Functional Actor: Admin Requirement: The system will allow admin to set the priority using 3 options (High, mid, low) Justification: The core function of the system Priority: High Originator: Developer Team Description: As Admin I need to assign priority to a ticket based on its content Fir Criterion: adding priority to a ticket means that the department/agent will know how important a ticket is. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 19, 2021 (Creation Date)

FR007	Type: Functional Actor: Admin Requirement: System will allow admin to add/edit date and time to tickets Justification: Core Function of system Priority: High Originator: Developer Team Description: As an admin, I should be able to add time and date to a ticket Fir Criterion: Adding Time and date to a ticket will allow for tracking and monitoring Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 20, 2021 (Creation Date)
FR008	Type: Functional Actor: Admin Requirement: Description of issue/issues. Justification: Core Function of system Priority: High Originator: User Description: As a user, I should be able to edit add/edit descriptions of issues for the ticket Fir Criterion: Adding a description box to a ticket will allow for additional information to be added Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 20, 2021 (Creation Date)

FR009	Type: Functional Actor: Admin Requirement: Add/edit the level of IT support. Justification: Core Function of system Priority: High Originator: Developer Team Description: As an admin, I should be able to add/edit the level of support required Fir Criterion: Adding a level of IT support requirements to a ticket will paint a better picture of the required support needed to resolve the ticket. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 20, 2021 (Creation Date)
FR010	Type: Functional Actor: User Requirement: Add contact details to tickets during submission. Justification: Core Function of system Priority: High Originator: Developer Team Description: As a user, I can add my contact details to the ticket upon creation Fir Criterion: Allowing the user to add contact details allows for better communication channels between user and Admin Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's

	History: October 20, 2021 (Creation Date) Add contact details to tickets during submission.
FR011	Type: Functional Actor: Admin Requirement: Add contact details to tickets during submission. Justification: Core Function of system Priority: High Originator: Developer Team Description: As an admin, I can add my contact details from the user to the ticket upon creation. Fir Criterion: Allowing the administrator to add contact details allows for better communication channels between user and Admin Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 20, 2021 (Creation Date) Categories for channels.
FR012	Type: Functional Actor: Admin Requirement: Categories for channels. Justification: Core Function of system Priority: High Originator: Developer Team Description: As an admin, I can assign a category to the ticket. Fir Criterion: Allowing the administrator to add a category to the ticket for better filtration of the ticket to ensure the ticket goes to the best-suited department. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's

	History: October 20, 2021 (Creation Date) Add final comments/reviews of performance.
FR013	Type: Functional Actor: User Requirement: Add final comments/reviews of performance. Justification: Core Function of system Priority: High Originator: Developer Team Description: As a user, I can provide feedback on how the ticket was handled. Fir Criterion: Allowing the user to provide feedback through a 5-star rating system will be a quantifiable measurement of performance. Conflicts: Nil Supporting material: Refer to the list of FR's and NFR's History: October 20, 2021 (Creation Date)

User Requirements

The system should allow adding the satisfaction of feedback and resolution of a submitted incident. The classification is from 1 to 5 stars.

System Requirements

Upon completion of ticket resolution, the user will be able to submit feedback on the overall experience.

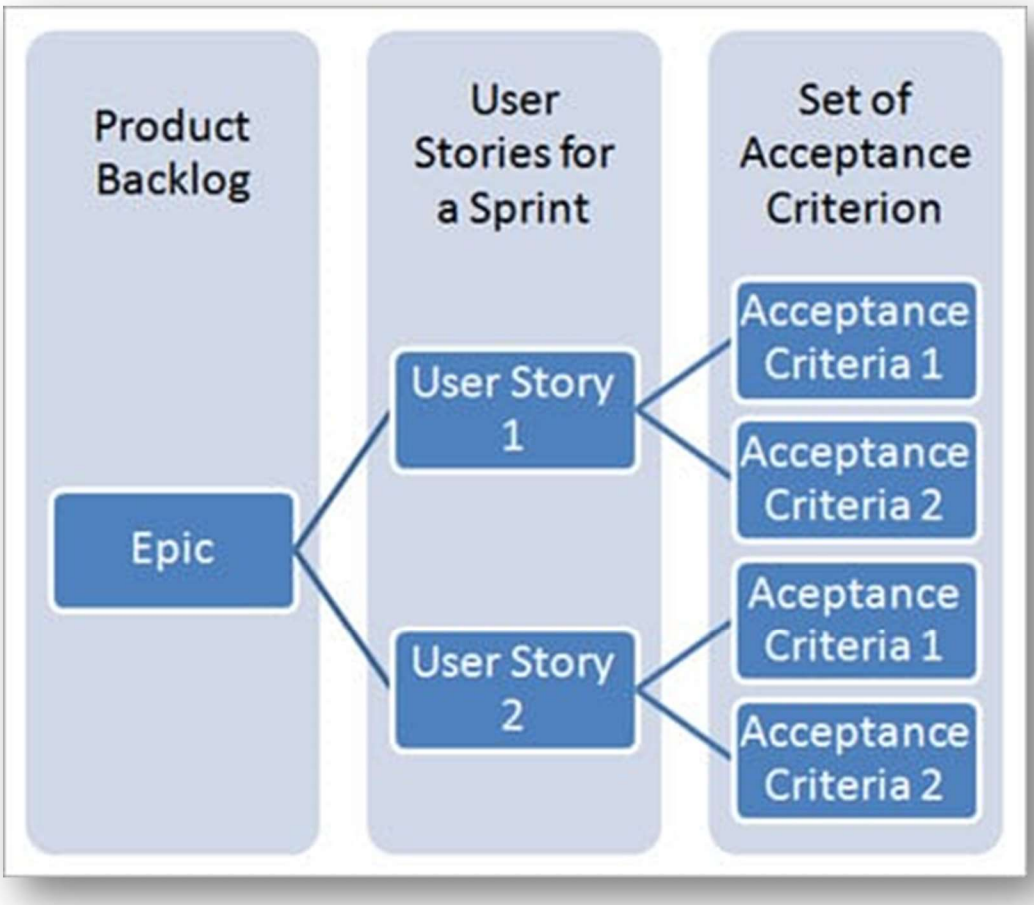
End-user can assign a star rating to go with feedback.

Administration can assess the feedback and rating to feed into analytics such as KPIs and reporting.

Monthly meeting on reports and feedback to improve internal systems.

User stories and scenarios.

1. Tom is a student at Yoobee College and is having mouse issues; The Right Click is sometimes not functioning at all.
2. Mike is a student at Yoobee College and needs Qt Software installed on his local student machine; Permissions will not allow Mike to make the needed changes himself.
3. Luke is a student at Yoobee college he has turned up to class today and has found his chair is broken. Luke needs a replacement chair.
4. Sarah is a new student at Yoobee, and she needs a new user account within the system.



(https://www.softwaretestinghelp.com/user-story-acceptance-criteria/,n.d)

User story and a scenario

1. Incident Logging

Tom is a student at Yoobee college. He is having an issue with his mouse where the right-click button will inconsistently work. Tom Emails Support to action the issue for resolution.

2. Ticket Creation

A ticket is generated by a Yoobee operating ticketing program via internet submission by Yoobee Administration.

3. Incident Categorization

Yoobee Administration has categorized based on the area of IT or business that the incident disrupts is Hardware related.

4. Incident Prioritization

The priority of an incident has been determined as a function impact. The incident has an impact on Tom's ability to use the current PC that he is on. The urgency of an incident indicates the time within which the incident should be resolved. Based on the priority, the incident is categorized as:

- Critical

5. Incident Resolution

The incident is considered resolved when the technician has provided a temporary workaround moving to another computer until a new mouse has replaced the non-functional one.

6. Incident Closure

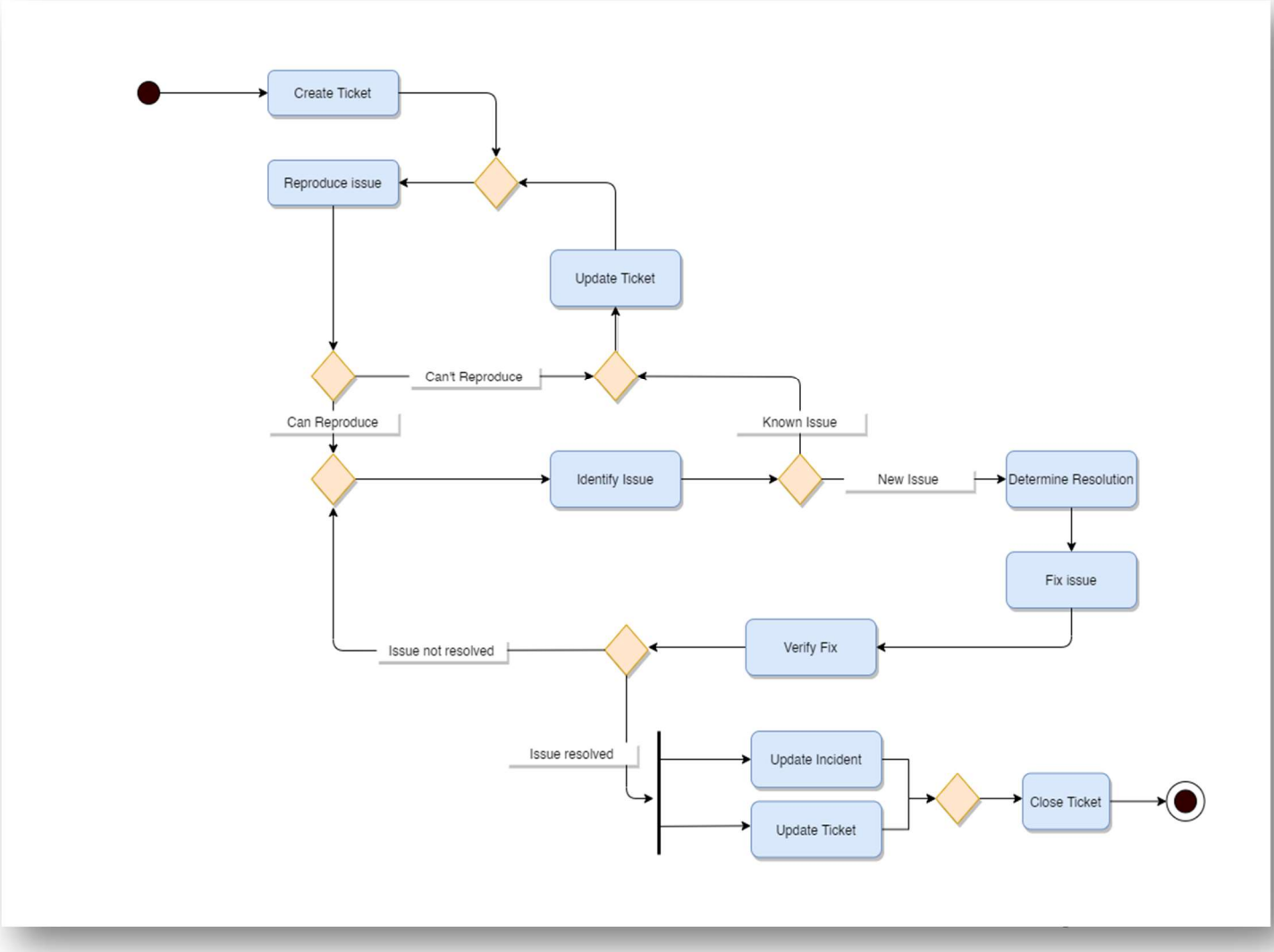
The incident has been closed once the issue is resolved and the user acknowledges the resolution and is satisfied with it.

Requirement prioritization and negotiation.

Priority has been outlined in the SRS above. The functional requirements meet a high prioritisation as they are key functions that all feed into the system. The system operates and a complete entity, thus high prioritization across the board.

System Modelling and Design.

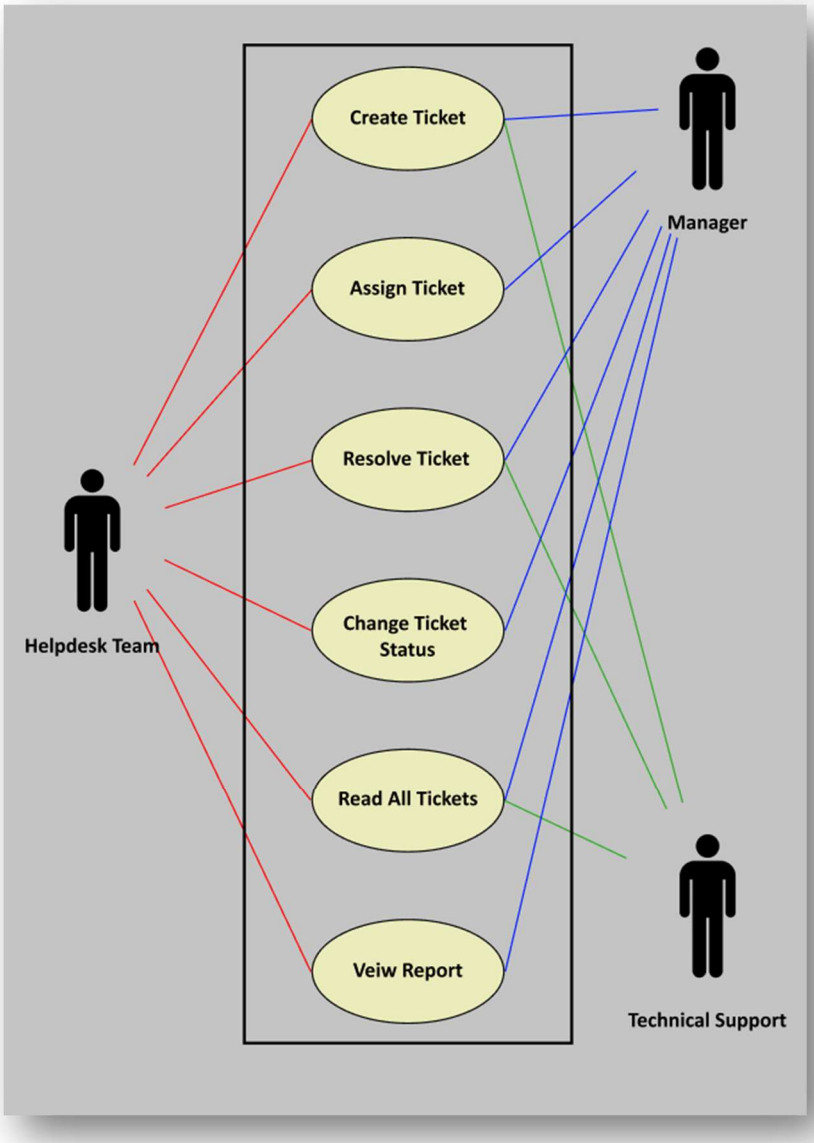
Activity Diagram.



Impacts

If there is any issue in the functionality of the project, we can track it by seeing the use-case and activity diagram. For example, if there is an error in the validation process, we can rectify it by seeing the activity diagram which gives us the complete details of the error.

Use case Diagram.



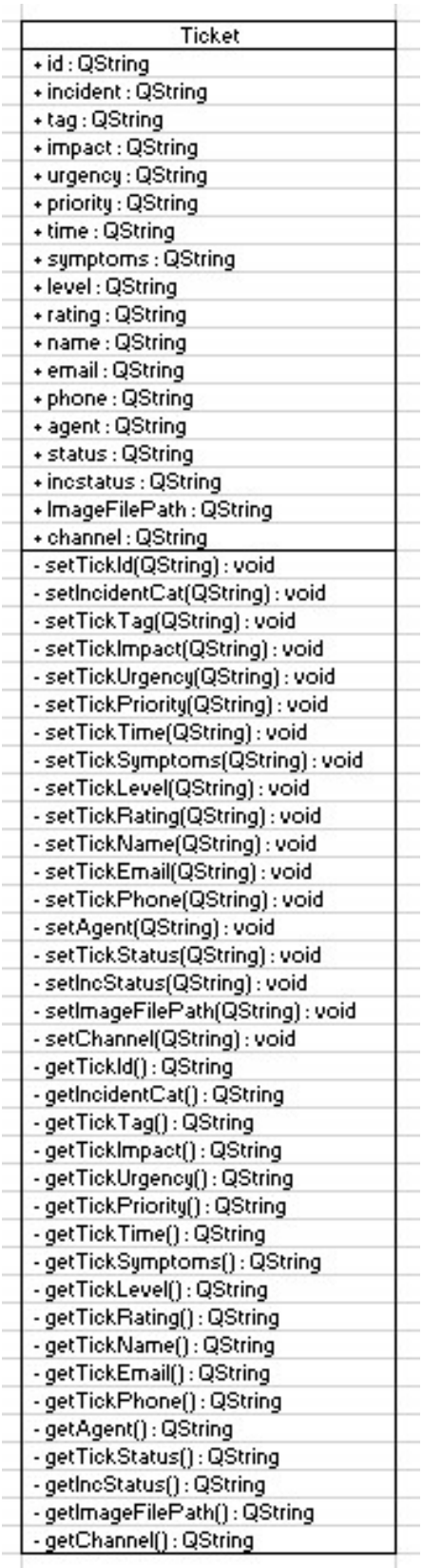
Helpdesk Team: This module has the entire access to all other modules, admin creates the project and assigns the projects created to the manager, adding members to the project, assigning defects based on the priority. It can update the manager, members, and access to the project data. Generating reports based on the managers' report submission.

Manager: This module has all administrative features to access once the role is assigned by an administrator.

Reports: The helpdesk team or Manager can access this module and generate the reports based on the requirements.

Technical Support: Can access the task or Defect assigned by the manager, view assigned projects, and resolve the assigned Defect. The developer can view the Defects list assigned by the manager.

Class Diagram.



Note:

Class diagram 'Ticket' shows the operations managed all under one class.

10 Class attributes and 21 methods can be seen in this class diagram

UI Design.

Screen Layout.

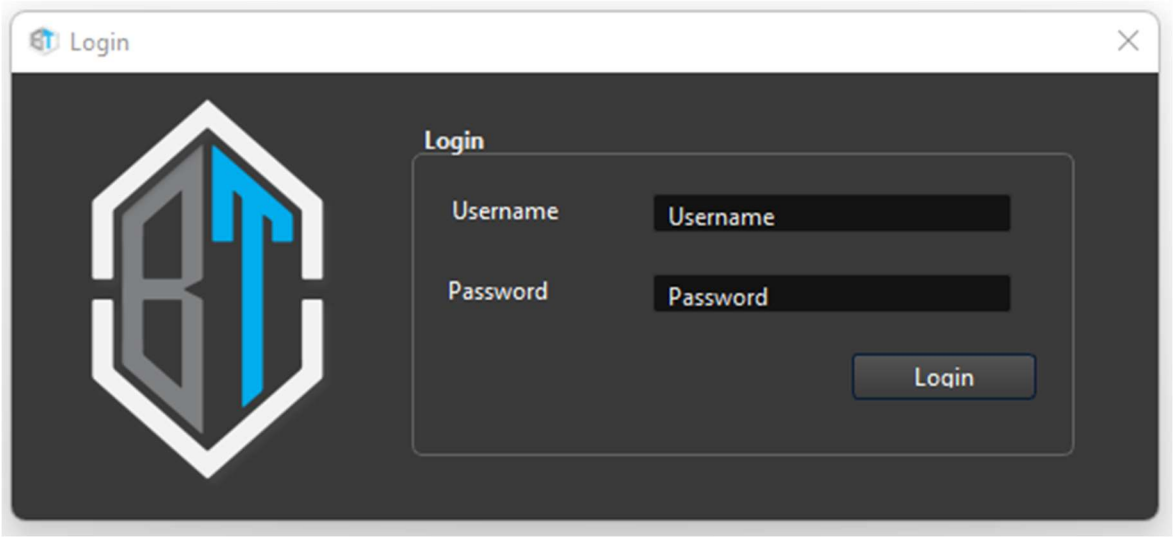
Low fidelity UI designs

Main and secondary windows.


Dashboard
Dashboard 2

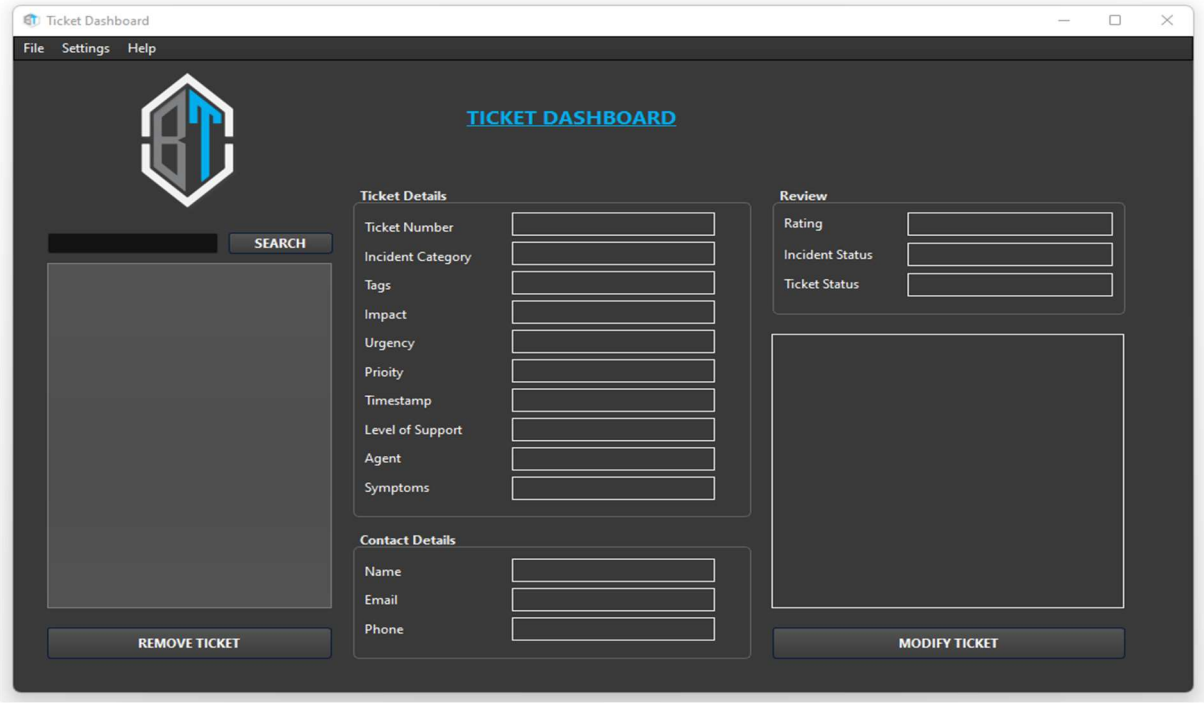
Prototypes & Form elements

Login Page:

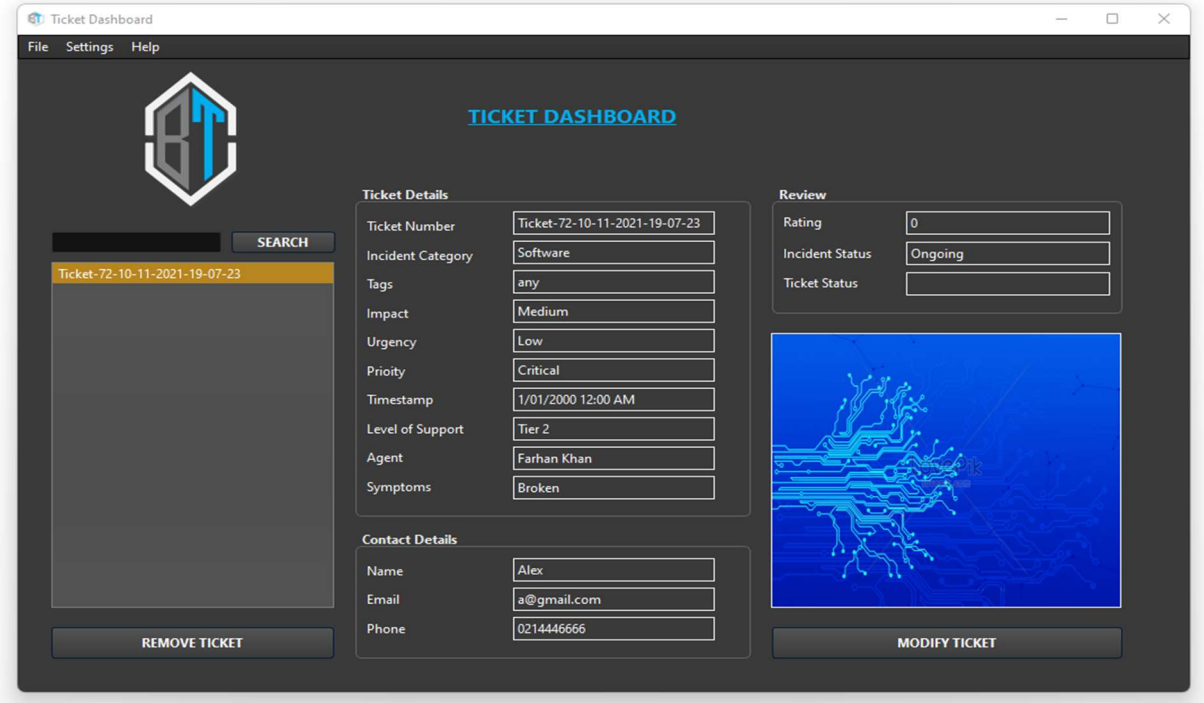


Basic Login screen for the administration, under the assumption that the program has been set up with login already in place. Form elements used are shown in the above image.

Dashboard Dialog: Unpopulated

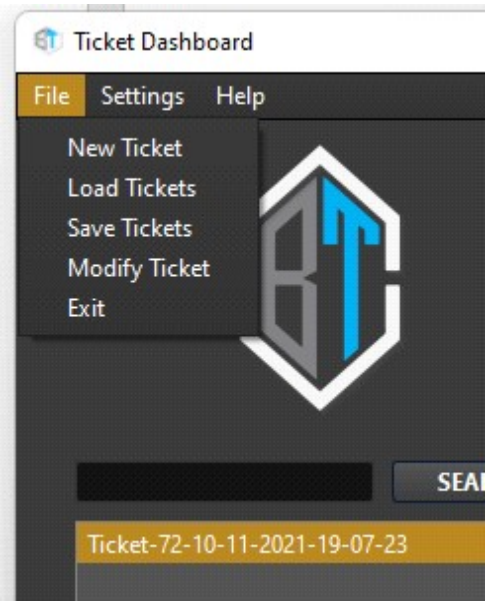


Dashboard Dialog: Populated

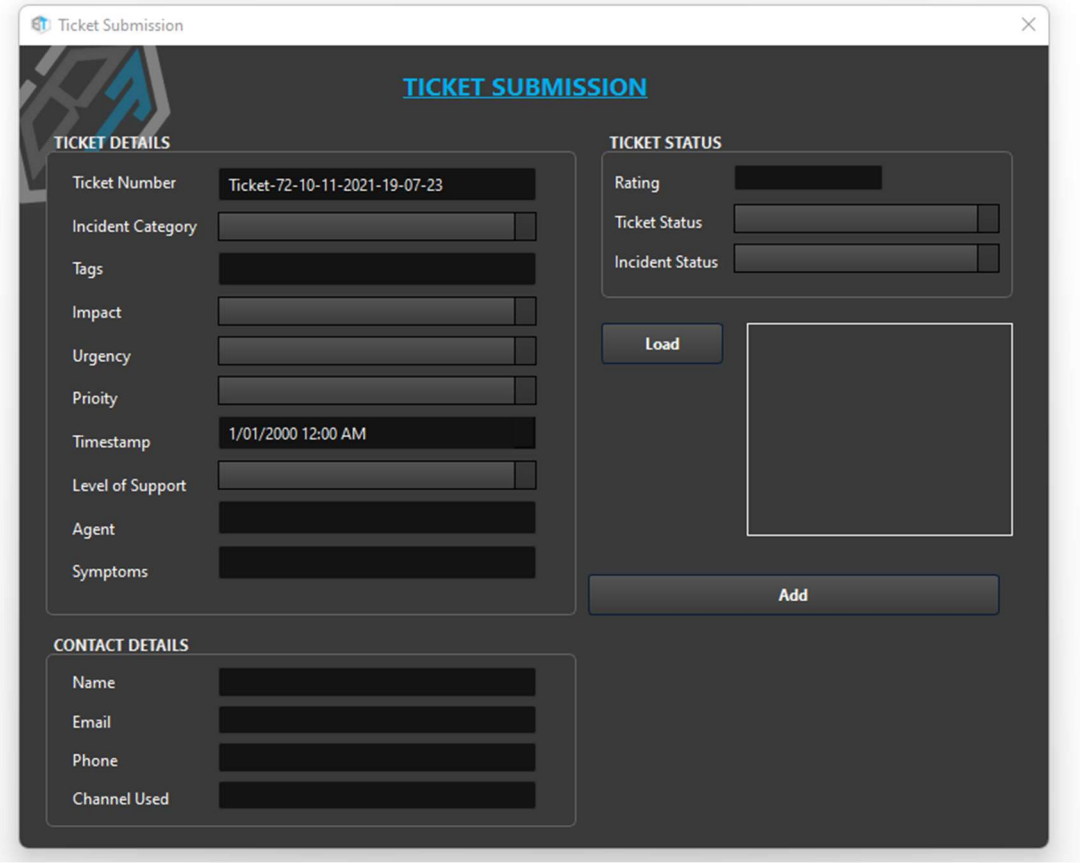


A dashboard dialog is a place for the system to display information about all tickets including a graph. The dialog will have the ability to remove tickets and allow the opening of 2 new dialogs known as edit and new.

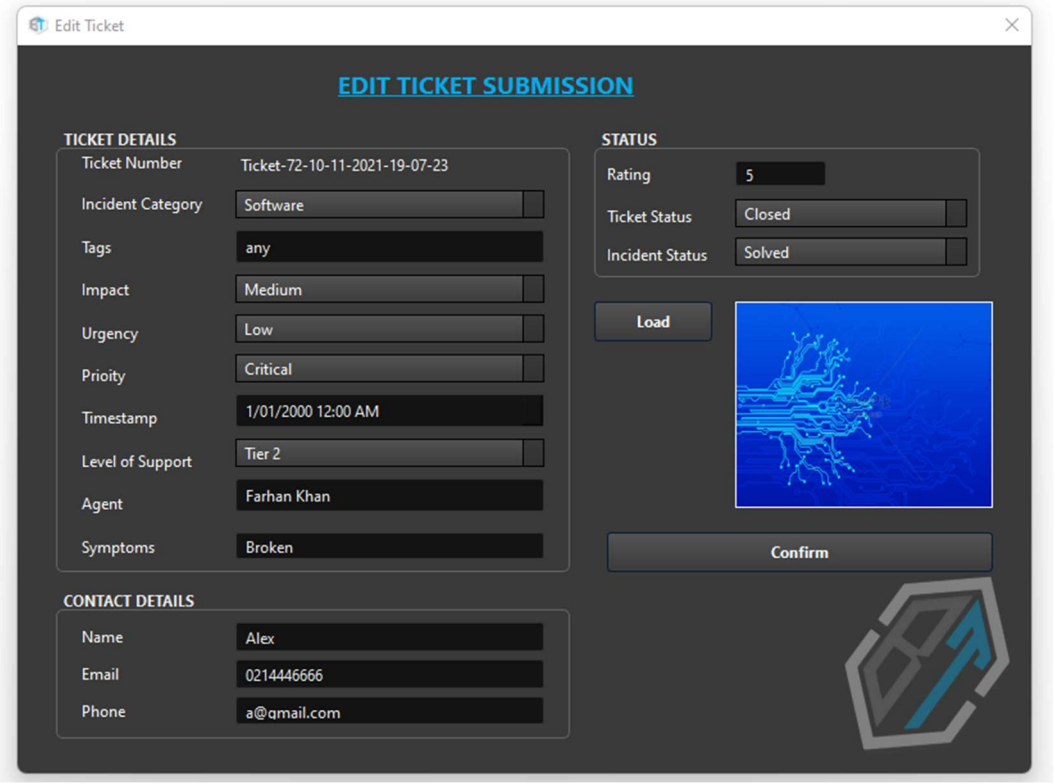
Edit Ticket Dialog:



Ticket Submission: Unpopulated



Ticket Submission: Populated



The system will use a mixture of class elements throughout the program.

These include:

1. Labels
2. Combo boxes
3. Horizontal Slider
4. Pushbuttons
5. Checkboxes
6. Line edits
7. Time edits
8. Date edits
9. Spinbox
10. List widget
11. Group box

Implementation (Coding).

Event handling code.

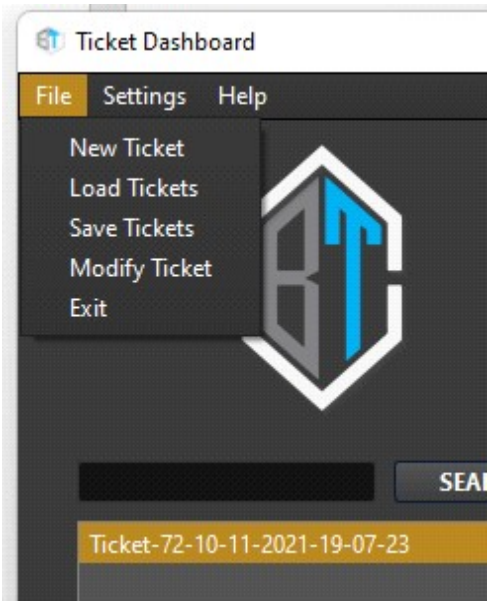
	Events	Function	Description
1	Login clicked	LoginDialog::on_pushButton_clicked()	Checks credentials, reads and loads previous saved tickets from designated txt files
2	New Ticket clicked	handleMenuTicketNew()	Opens addticketdialog, preloads some allocated default values set by index
3	New Load Image button clicked	loadItemImage()	Check & create QDir().mkdir("./images") Save file using image file path and name
4	New Ticket Add button clicked	confirmAdd()	Checks if fields are empty using if statements saves input and passes strings to UI and vector
5	Load Tickets clicked (Qaction)	handleLoadTickets()	Checks for file location/opens txt file/ reads and passes data to UI labels
6	Save Tickets clicked (Qaction)	handleSaveTickets()	Checks for file location/opens txt file/ writes and passes data to txt file
7	Removed button clicked	removeSelectedTicket()	Reads users selection from list and removes data from Ui and txt file
8	Modify Tickets clicked (Qaction)	handleMenuTicketEdit()	Opens editicketdialog and load users selected ticket data from file
9	Modfiy button clicked	handleMenuTicketEdit()	Passes new data to txt file (apend) and to Ui labels in main window
10	Modify add button clicked	confirmUpdate()	Checks if fields are empty using if statements, apends txt file and passes strings to UI and vector
11	Modify load button clicked	loadItemImage()	Checks filename is != "" copys selected image to build folder set pixmap to ui labels in main window
12	Search button clicked	on_pushButton_clicked()	Allows user to enter data into line edit and execute a search within the List widget returns a red colour to highlight ticket
13	List ticket selected	handleTicketClick()	Checks if index != -1, highlights ticket list and sets and gets Ui label data related to ticket from vector
14	Lightmode clicked	handleThemeLightmode()	Opens qss file, reads and over writes qt stylesheet
15	Darkmode clicked	handleThemeDarkmode()	Opens qss file, reads and over writes qt stylesheet
16	Exit clicked (Qaction)	handleMenuExit()	Close() closes program

Signals and Slots.

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    connect(ui->actionNew, &QAction::triggered, this, &MainWindow::handleMenuTicketNew);
    connect(ui->actionLoad, &QAction::triggered, this, &MainWindow::handleLoadTickets);
    connect(ui->actionSave, &QAction::triggered, this, &MainWindow::handleSaveTickets);
    connect(ui->actionExit, &QAction::triggered, this, &MainWindow::handleMenuExit);
    connect(ui->pb_remove, &QPushButton::clicked, this, &MainWindow::removeSelectedTicket);
    connect(ui->lst_tickets, &QListWidget::itemClicked, this, &MainWindow::handleTicketClick);
    connect(ui->pb_modify, &QPushButton::clicked, this, &MainWindow::handleMenuTicketEdit);
    connect(ui->actionModify_Ticket, &QAction::triggered, this, &MainWindow::handleMenuTicketEdit);
    connect(ui->pushButton, &QPushButton::clicked, this, &MainWindow::searchProduct);
    connect(ui->actionSpyBot, &QAction::triggered,this, &MainWindow::handleThemeDarkmode);
    connect(ui->actionGravira, &QAction::triggered,this, &MainWindow::handleThemeLightmode);

    handleLoadTickets();
    ui->lst_tickets->setCurrentRow(0);
    handleTicketClick();

}
```

```
addticketdialog::addticketdialog(Ticket*& newTicket, QWidget *parent) : QDialog(parent), ui(new Ui::addticketdialog)
{
    ui->setupUi(this);
    this->newTicket = &newTicket;
    imageFilePath = "none.png"; //default

    connect(ui->pushButton, &QPushButton::clicked,this, &addticketdialog::confirmAdd);
    connect(ui->pushButton_2, &QPushButton::clicked,this, &addticketdialog::loadItemImage);

    QDir pathDir("./images");
    if(!pathDir.exists())
    {
        //create it
        QDir().mkdir("./images");
    }

    int num = QRandomGenerator::global()->bounded(0,100);
    QDate date = QDate::currentDate();
    QTime time1 = QTime::currentTime();
    QString str3 = date.toString("dd-MM-yyyy");
    QString str4 = time1.toString("hh-mm-ss");
    QString str = QString::number(num);
    QString id = "Ticket-" + str + "-" + str3 + "-" + str4;

    ui->lineEdit_2->setText(id);
```

In the previous screenshots we can see that the key signals used are QAction::triggered, QListWidget::item clicked, and QPushButton::clicked.



QAction::triggered is related to the file drop down menu. This connects the actions such as 'New Ticket' with the slot i.e., 'do something'. In this case it is connected to the MainWindow function handleMenuTicketNew. This allows the admin to create a brand-new ticket.

QAction is used a total of 6 times in the program to allow user to interact with the ticketing function and to change the theme.

QPushButton::clicked and QListWidget::itemclicked are similar as they are both clickable widgets. This means that the signal is sent once clicked and connects with the slot (function).

Base classes and derived classes.

```
#ifndef TICKET_H
#define TICKET_H
#include <QString>
#include <QDate>

class Ticket
{
private:
    QString id;
    QString incident;
    QString tag;
    QString impact;
    QString urgency;
    QString priority;
    QString time;
    QString symptoms;
    QString level;
    QString rating;
    QString name;
    QString email;
    QString phone;
    QString agent;
    QString status;
    QString incstatus;
    QString ImageFilePath;
    QString channel;

public:
    Ticket( QString id, QString incident, QString tag, QString impact,
            QString urgency, QString priority, QString time, QString symptoms,
            QString level, QString rating, QString name, QString email,
            QString phone, QString agent, QString status, QString Incstatus, QString ImageFilePath, QString channel);

    void setTickId(QString id);
    void setIncidentCat(QString incident);
    void setTickTag(QString tag);
    void setTickImpact(QString impact);
    void setTickUrgency(QString urgency);
    void setTickPriority(QString priority);
    void setTickTime(QString time);
    void setTickSymptoms(QString symptoms);
    void setTickLevel(QString level);
    void setTickRating(QString rating =0);
    void setTickName(QString name);
    void setTickEmail(QString email);
    void setTickPhone(QString phone);
    void setAgent(QString agent);
    void setTickStatus (QString status);
    void setIncStatus(QString incstatus);
    void setImageFilePath(QString incstatus);
    void setChannel(QString channel);

    QString getTickId();
    QString getIncidentCat() const;
    QString getTickTag() const;
    QString getTickImpact() const;
    QString getTickUrgency() const;
    QString getTickPriority() const;
    QString getTickTime() const;
    QString getTickSymptoms() const;
    QString getTickLevel() const;
    QString getTickRating() const;
    QString getTickName() const;
    QString getTickEmail() const;
    QString getTickPhone() const;
    QString getAgent() const;
    QString getTickStatus() const;
    QString getIncStatus() const;
    QString getImageFilePath() const;
    QString getChannel() const;

};

#endif // TICKET_H
```

Stylesheets.

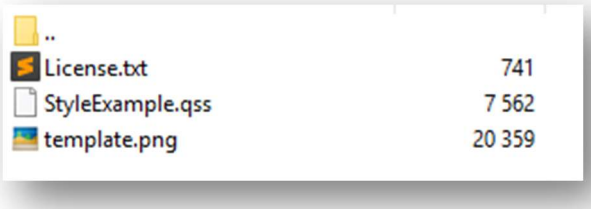
The use of stylesheets has streamlined the process in making the project visual look good. For the implementation of stylesheets originally used 1 stylesheet template downloaded from https://qss-stock.devsecstudio.com/templates.php where there is a large number of stylesheets to choose from for example:

(https://qss-stock.devsecstudio.com,n.d)

The process of adding the stylesheets was very simple to follow:

Download

You will have a file of the type .rar, an archive which bears the name of the chosen template (StyleEaxmple.rar). This archive contains:



-Style Example.qss : the file that contains the syntax of Qt Style Sheets customize and adaptable to all chosen graphic components.

-License.txt : contains the text of the license which authorizes the use of the file in the respect of the copyrights for the moment it is the license MIT.

-template.png : a preview on the result of the style sheets on the interface.

External reading from file :

```
#include "mainwindow.h"
#include <QApplication>
#include <QFile>

int main(int argc, char *argv[])
{
        //create the application and the main window
        QApplication app(argc, argv);
        MainWindow w;

        //open qss file
        QFile file("path/StyleExample.qss");
        file.open(QFile::ReadOnly);

        QString styleSheet { QLatin1String(file.readAll()) };

        //setup stylesheet
        app.setStyleSheet(styleSheet);

        //run
        w.show();

        return app.exec();

}
```

Data Model.

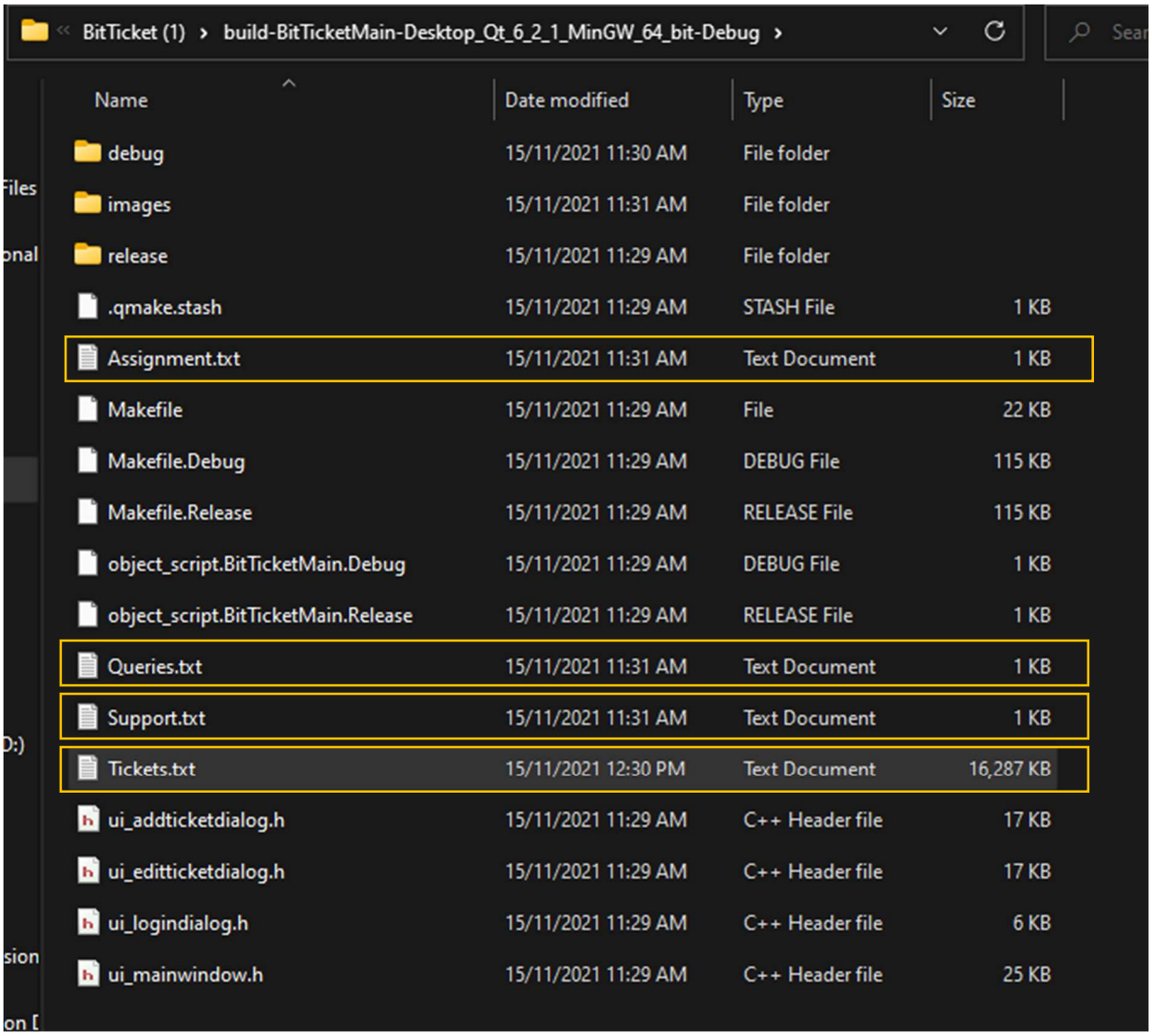
Int = Ticket Number.

String = Incident Category.

String = Tags.

- Impact.
- Urgency.
- Priority.
- Timestamp.
- Symptoms.
- Level of support.
- User Details who submitted the incident.
- User details on who registered the incident.
- The channel used to submit the incident.
- Additional information about the incident, and attachments.

File Management (I/O).



We have included 4 files for this program, The first major File that is utilized is the Tickets.txt file. This file is the main save file for all ticket information to be stored. This file is loaded on the execution of the program and is displayed in the Ticket list widget.

The second file included is the assignment.txt file. The purpose of this file is to contain information that relates to a specific agent, which can be chosen during the add ticket dialog or the edit ticket dialog. At this point, the scope did not allow for further implementation for the file to be utilised. The principal idea was to allow for data to be captured in a new file for notification to the agent.

The supported file relates to the assignment file as data is logged based on the tier allocated at the new ticket stage. Again because of the scope, no UI implementation has been added to the system.

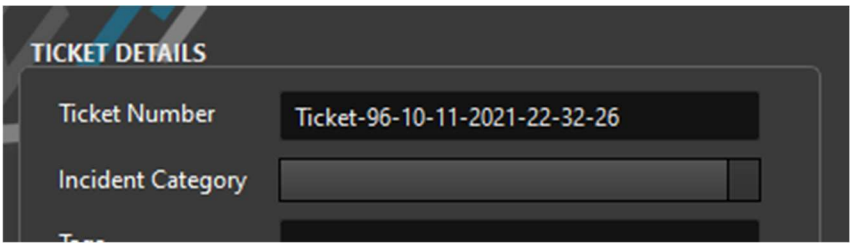
The final file is the queries.txt file. This file is like the ticket file as it logs the data from the system and in theory will generate a message about the logged ticket.

Software Testing.

In the following section, records have been detailed to show the testing and evaluation of the system and its functions. These tests are designed to determine if the SRS has been fulfilled.

Unit and system tests

Unit testing has been undertaken to see if the class operates as designed. The system tests are to assess if the functionality of the system and its functions work in parallel. The following sections detail the tests and results.

Test Case: TC1
Purpose: Auto allocation of a number to a ticket. **Precondition:** QString getTickId(); **Inputs**: QString id = ui->lineEdit_2->text(); **Expected Outcome:** This number is automatically generated and is displayed below   

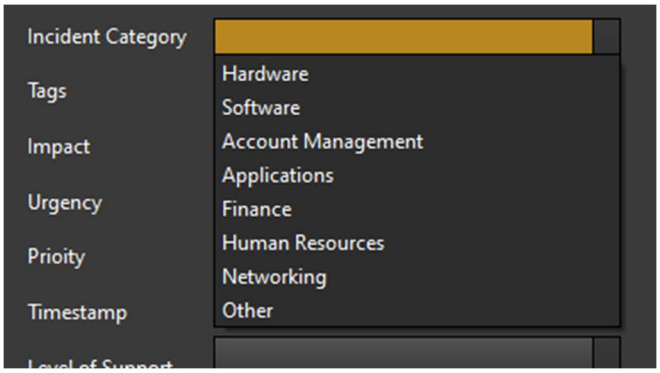
Test Case: TC2

Purpose: Assign the ticket to the department.

Precondition: QString getIncidentCat() const;

Inputs: QString incident = ui->comboBox_7->currentText();

Expected Outcome: Generation with a combo box display of the below items



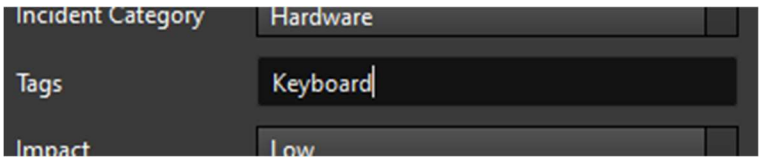
Test Case: TC3

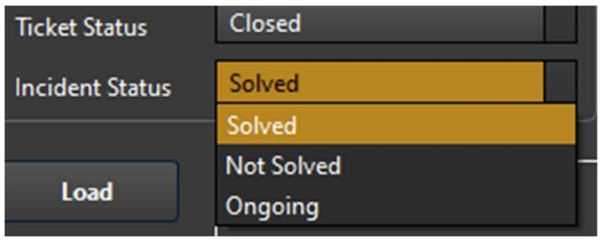
Purpose: Add/edit tags to the ticket

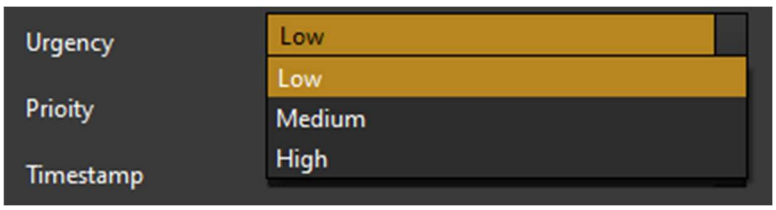
Precondition: QString getTickTag() const;

Inputs: QString tag = ui->lineEdittags->text();

Expected Outcome: A line edit is presented for the user to add an entry



Test Case: TC4
Purpose: Allow the admin to determine outcomes **Precondition:** QString getIncStatus() const; **Inputs:** QString incstatus = ui->comboBox_IS->currentText(); **Expected Outcome:** Generation with a combo box display of the below items   

Test Case: TC5
Purpose: Admin to set the urgency of the ticket **Precondition:** QString getTickUrgency() const; **Inputs:** QString urgency = ui->comboBox_10->currentText(); **Expected Outcome:** Generation with a combo box display of the below items   

Test Case: TC6

Purpose: Admin to set the priority

Precondition: QString getTickPriority() const;

Inputs: QString priority = ui->comboBox_11->currentText();

Expected Outcome: Generation with a combo box display of the below items

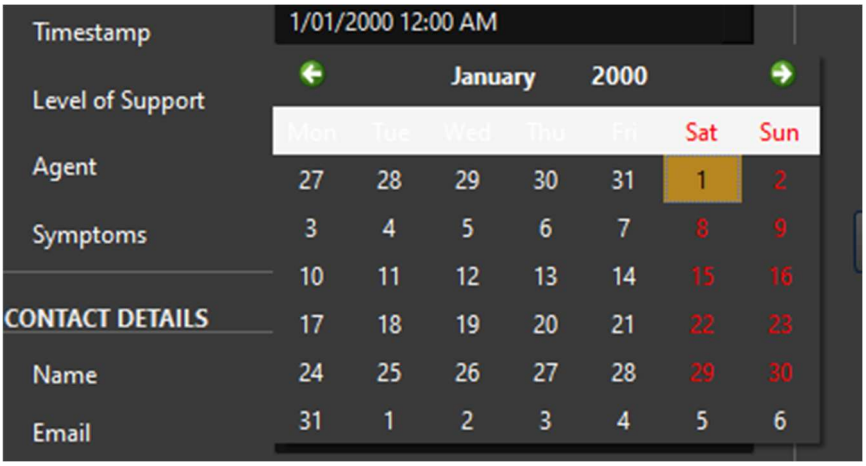
Test Case: TC7

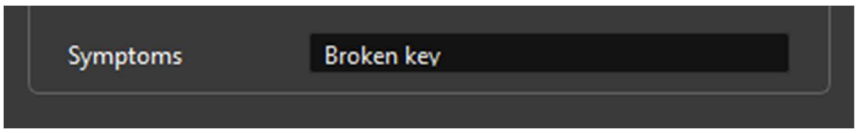
Purpose: Add/edit dates and times to tickets

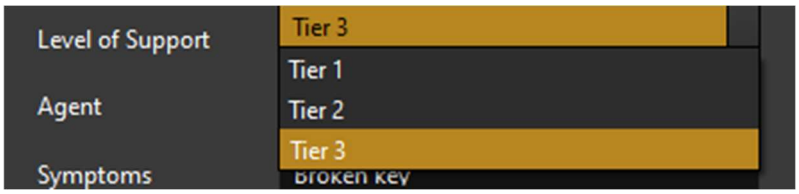
Precondition: QString getTickTime() const;

Inputs: QString time = ui->dateTimeEdit_2->text();

Expected Outcome: Generation with a Calendar view display below



Test Case: TC8
Purpose: Describe issues for the admin to assess. **Precondition:** QString getTickSymptoms() const; **Inputs:** QString symptoms = ui->txtSymptoms1->text(); **Expected Outcome:** A line edit is presented for the user to add an entry 

Test Case: TC9
Purpose: Add/edit the level of IT support or corresponding department hierarchy **Precondition:** QString getTickLevel() const; **Inputs:** QString level = ui->comboBox_12->currentText(); **Expected Outcome:** Generation with a combo box display of the below items 

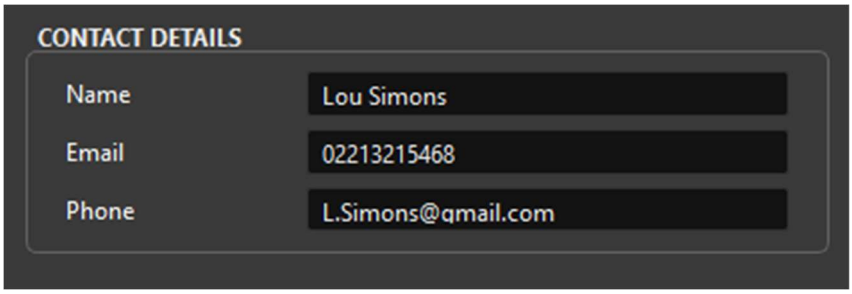
Test Case: TC10

Purpose: Add contact details to tickets during submission.

Precondition: QString getTickName() const;
QString getTickEmail() const;
QString getTickPhone() const;

Inputs: QString name = ui->lineEdit_13->text();
QString email = ui->lineEdit_14->text();
QString phone = ui->lineEdit_15->text();

Expected Outcome: Line edits is presented for the user to add an entry



Test Case: TC11

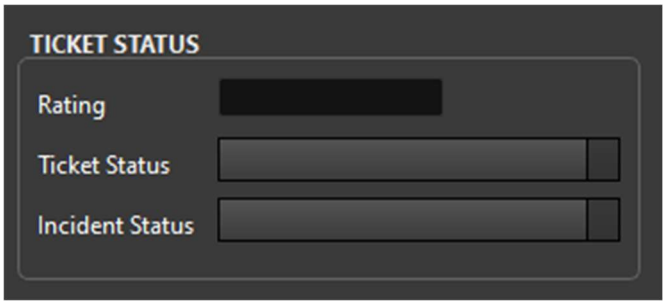
Purpose: Add contact details to tickets during submission.

Precondition: QString getTickName() const;
QString getTickEmail() const;
QString getTickPhone() const;

Inputs: QString name = ui->lineEdit_13->text();
QString email = ui->lineEdit_14->text();
QString phone = ui->lineEdit_15->text();

Expected Outcome: Line edits is presented for the user to add an entry

Test Case: TC12
Purpose: Categories for channels to record how the ticket was submitted. **Precondition:** QString getChannel() const; **Inputs:** QString channel = ui->txtRating_2->text(); **Expected Outcome:** A line edit is presented for the user to add an entry 

Test Case: TC13
Purpose: Final comments/reviews of performance. **Precondition:** QString getTickRating() const; **Inputs:** QString rating = ui->txtRating->text(); **Expected Outcome:** This has been updated now to a combo box with ratings from ‘0’ to ‘5’ 

Test Results & Evaluation

Test Case	Pass/Fail	Comments
TC1	Pass	
TC2	Pass	
TC3	Pass	
TC4	Pass	
TC5	Pass	
TC6	Pass	
TC7	Pass	
TC8	Pass	
TC9	Pass	
TC10	Pass	
TC11	Pass	
TC12	Pass	
TC13	Pass	

Release Testing.

The primary goal of the release testing process is to showcase a working system to ensure that the system is good for use.

As part of release testing, the development team broke down system testing into small bite-sized sections while developing the program. This allowed us to keep on top of any issues that occurred.

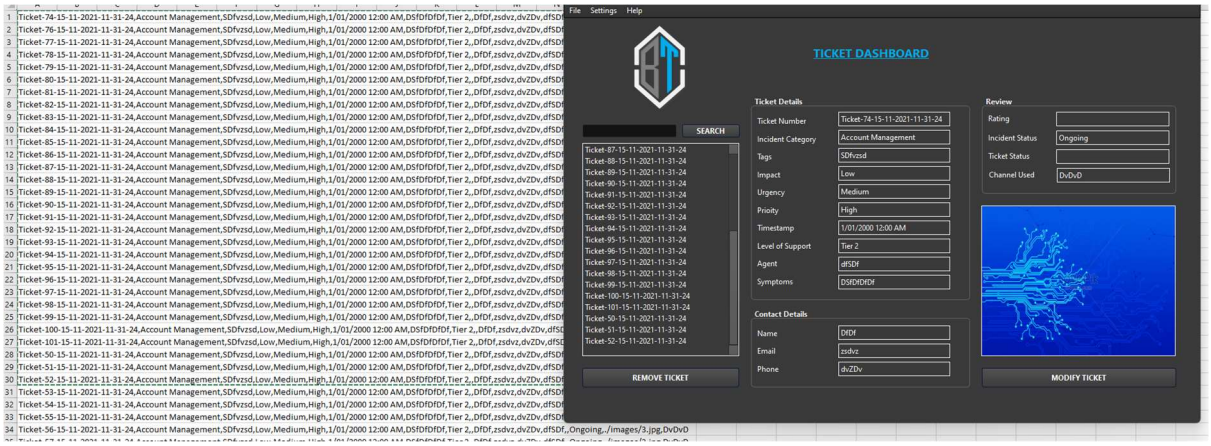
Early testing found that we had issues with the login page. The program work for all purposes until the user clicked the close button in the top right-hand corner. This shut down the whole program without saving any ticket information. The fix for this was to run a second dialog as the dashboard and main window dialog for system login.

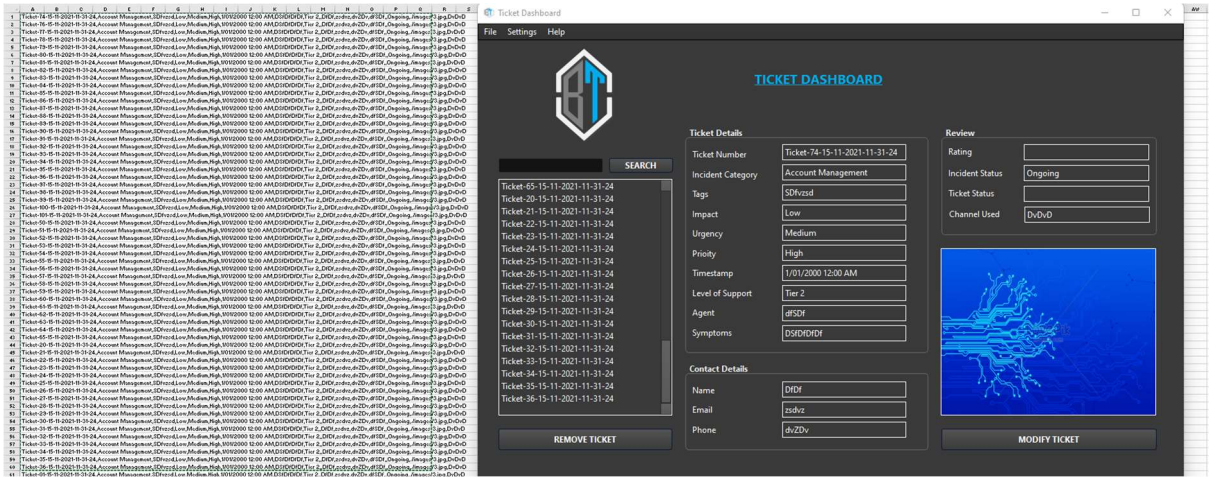
The system was bug-free, we did come across other minor bugs that such as Qss files for the UI design seemed to interfere with the program functionality, i.e., the ability to open combo boxes and interact with some of the line edits. The fix was to remove background colors set to transparent.

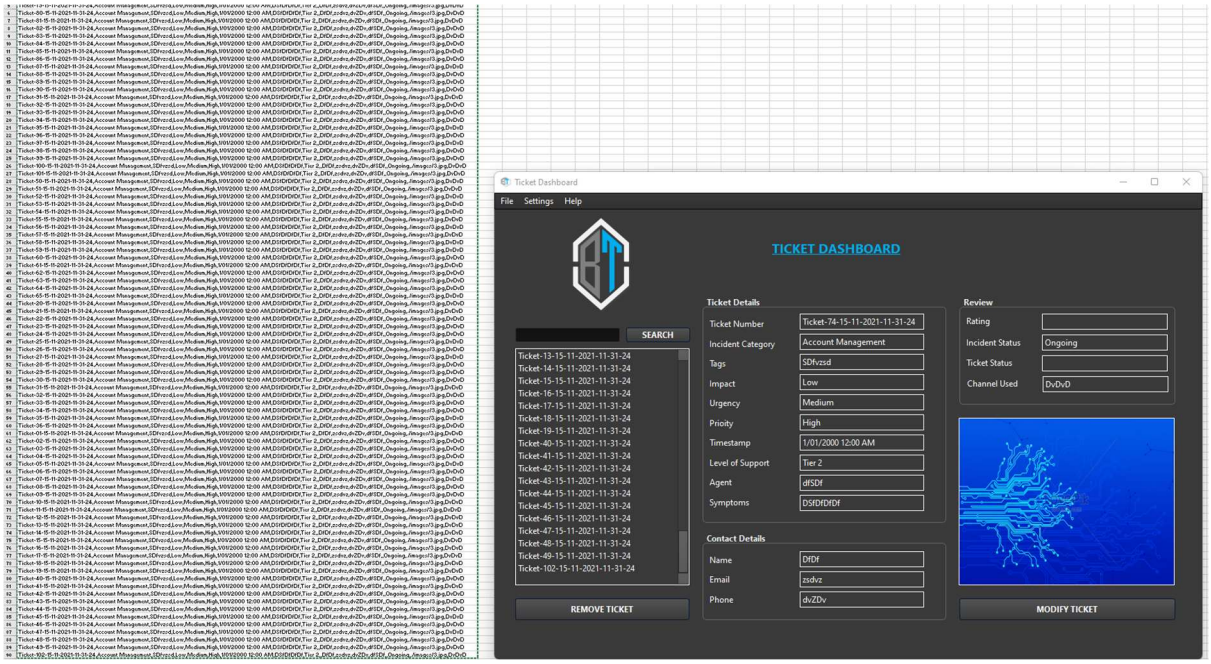
QSS files also presented issues when trying to implement light mode and dark mode. Some minor adjustments to the file locations seem to fix this until we came to relaunch the program in the following days. The ultimate fix was to incorporate the Qss in the resource file and set the file path to reflect that.

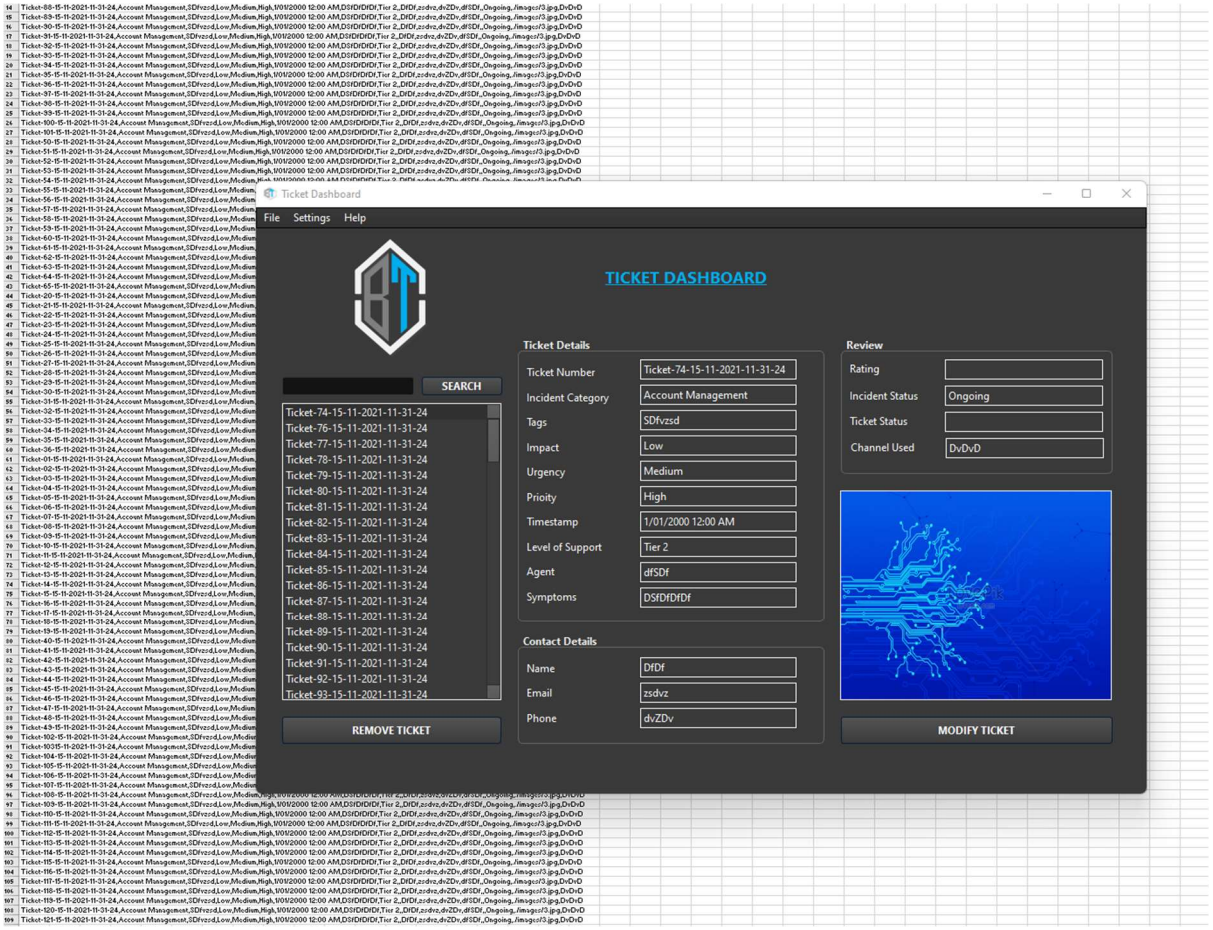
Performance testing:

Realistically the only loading that could be applied to the program was the volume of tickets. The first round of testing began with loading and saving tickets in increments of 30.





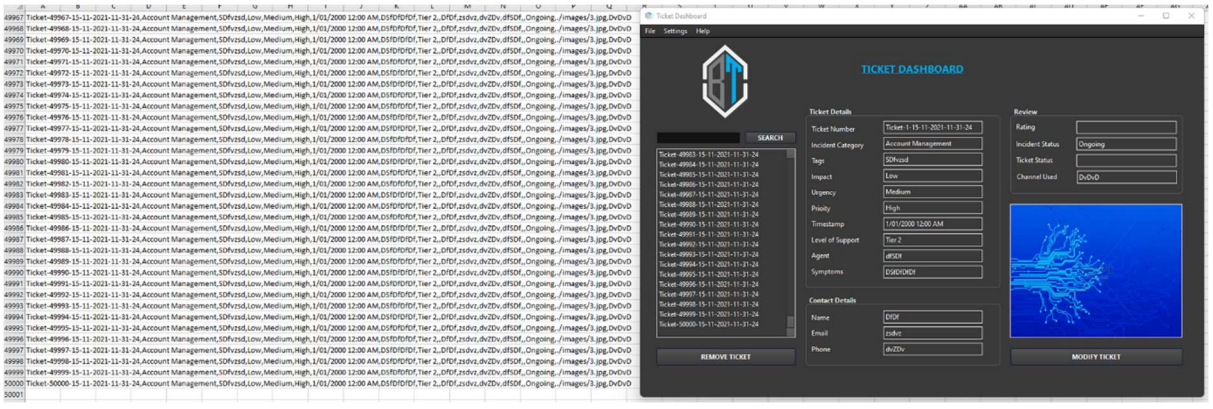




As can be seen above each stage of ticket incrementation at 30, 60, and 90 tickets. All were successful.

The final ticket volume was 109 with no problems. At 100-tickets, manual editing of the text file using excel to number and concatenate large amounts of tickets meant that we didn't have to manually enter 100+ tickets.

Curiosity drove the team to try and crash the program by increasing the volume exponentially. Testing the program to load 50,000 tickets returned surprising results.



At 50,000 tickets the program loading time increase slightly but did not affect the performance of the system once loading was complete.

Alpha Testing.

During the Alpha Testing with the client (Farhan Khan from Yoobee) some discussion around the Ticket display upon program execution. Coding was implemented to allow for the auto display of tickets.

```
handleLoadTickets();
ui->lst_tickets->setCurrentRow(0);
handleTicketClick();
```

The client also suggested having a rating displayed and editable during the submission, at that time the rating was only editable within the edit ticket dialog. The ticket rating combo box has been added to the add ticket dialog to allow the admin/user to adjust.

Beta Testing.

Tester: Tess

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Easy	10/11/2021	Pass	Opened Username and Password.vbs file and input matching Login and Password to BitTicket App
Generate a Ticket	Easy	10/11/2021	Pass	File > New Ticket > Completed Details > Added
Generate a Second Ticket	Easy	10/11/2021	Pass	Same as above
Search on any Ticket	N/A	10/11/2021	Fail	Input partial ticket number then full ticket number, neither search highlighted matching ticket
Locate a number to a ticket.	Easy	10/11/2021	Pass	Located in the first line of Ticket Details
Edit tags to the ticket.	Easy	10/11/2021	Pass	Selected Ticket > Modify Ticket > Edited Tag > Confirmed
Set the urgency of the ticket	Easy	10/11/2021	Pass	Completed through adding a new ticket or modifying an existing one.
Set the priority using 3 one of the 3 options	Easy	10/11/2021	Pass	Completed through adding a new ticket or modifying an existing one.
Modify the dates and times for tickets.	N/A	10/11/2021	Fail	Was able to change the date/time in modification but the timestamp didn't change in the Ticket Detail display.
Edit the level of IT support or corresponding department hierarchy.	Easy	10/11/2021	Pass	Completed through adding a new ticket or modifying an existing one.
add contact details to tickets during submission.	Easy	10/11/2021	Pass	Input Name, Email, & Contact Number
Modify categories for channels to record how the ticket was submitted.	N/A	10/11/2021	Fail	Can only input channel when adding a ticket, but unable to view or modify once added.

Additional Notes:
The File drop-down menu contains a non-functional Modify Ticket

The light and dark mode is an awesome addition! The only issue is when on light mode, the Theme options are difficult to see while changing back.

Tester: Nolene

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	Easy
Generate a Ticket	Medium	10/11/2021	Pass	Easy - date stamp not working - no rules as far as I can pick up since I tried to break it. Dropdown is good
Generate a Second Ticket	Complex	10/11/2021	Pass	Generated
Search on any Ticket	Complex	10/11/2021	Fail	Not working
Locate a number to a ticket.	Medium	10/11/2021	Pass	not sure what you want from me - didn't do anything
Edit tags to the ticket.	Medium	10/11/2021	Pass	Easy - Pass
Set the urgency of the ticket	Complex	10/11/2021	Pass	is this through the update you want me to do? No action
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	imagine myself tilting my head here - caus you crazy talking
Modify the dates and times for tickets.	Medium	10/11/2021	Pass	modify ticket - easy
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Blocked	modify ticket?
add contact details to tickets during submission.	Medium	10/11/2021	Pass	new ticket?
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	??

Tester: Julie

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	
Generate a Ticket	Medium	10/11/2021	Pass	
Generate a Second Ticket	Complex	10/11/2021	Blocked	
Search on any Ticket	Complex	10/11/2021	Fail	Nothing on the screen changes when I search so it's hard to tell whether it searched since it's already loaded in after you submit a ticket. Update: added another ticket to see if that was what I was missing, still no noticeable change when searching sorry.
Locate a number to a ticket.	Medium	10/11/2021	Maybe?	Not exactly sure what this means
Edit tags to the ticket.	Medium	10/11/2021	Pass	
Set the urgency of the ticket	Complex	10/11/2021	Pass	
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	
Modify the dates and times for tickets.	Medium	10/11/2021	Fail	
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Pass	
add contact details to tickets during submission.	Medium	10/11/2021	Pass	
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	

Tester: Mike

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	Very Easy yo do with the vb script
Generate a Ticket	Medium	10/11/2021	Pass	Easy to do, Nice interface
Generate a Second Ticket	Complex	10/11/2021	Pass	Just as easy as the first one
Search on any Ticket	Complex	10/11/2021	Fail	The search wasn't working
Locate a number to a ticket.	Medium	10/11/2021	Pass	Easy
Edit tags to the ticket.	Medium	10/11/2021	Pass	Easy with modifying, only file > modify wasn't working. The button was
Set the urgency of the ticket	Complex	10/11/2021	Pass	Also, an easy task with a nicely laid out interface.
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	Setting priority was easy
Modify the dates and times for tickets.	Medium	10/11/2021	Blocked	UI has a hidden date clicker.
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Pass	Modify menu was great how everything was already auto-filled.
add contact details to tickets during submission.	Medium	10/11/2021	Pass	Easy to add all details.
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	I enjoyed the modify button

Tester: Alex

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	
Generate a Ticket	Medium	10/11/2021	Pass	
Generate a Second Ticket	Complex	10/11/2021	Pass	
Search on any Ticket	Complex	10/11/2021	Fail	Can use the search box but nothing happens after the search is clicked
Locate a number to a ticket.	Medium	10/11/2021	Pass	
Edit tags to the ticket.	Medium	10/11/2021	Pass	
Set the urgency of the ticket	Complex	10/11/2021	Pass	
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	
Modify the dates and times for tickets.	Medium	10/11/2021	Blocked	Found the calendar but wasn't clear UI too dark
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Pass	
add contact details to tickets during submission.	Medium	10/11/2021	Pass	
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	

Tester: Daniel

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	
Generate a Ticket	Medium	10/11/2021	Pass	
Generate a Second Ticket	Complex	10/11/2021	Pass	
Search on any Ticket	Complex	10/11/2021	Fail	Didn't work
Locate a number to a ticket.	Medium	10/11/2021	Pass	
Edit tags to the ticket.	Medium	10/11/2021	Pass	
Set the urgency of the ticket	Complex	10/11/2021	Pass	
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	
Modify the dates and times for tickets.	Medium	10/11/2021	Fail	Didn't change the date or time
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Pass	
add contact details to tickets during submission.	Medium	10/11/2021	Pass	
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	

Tester: Jason

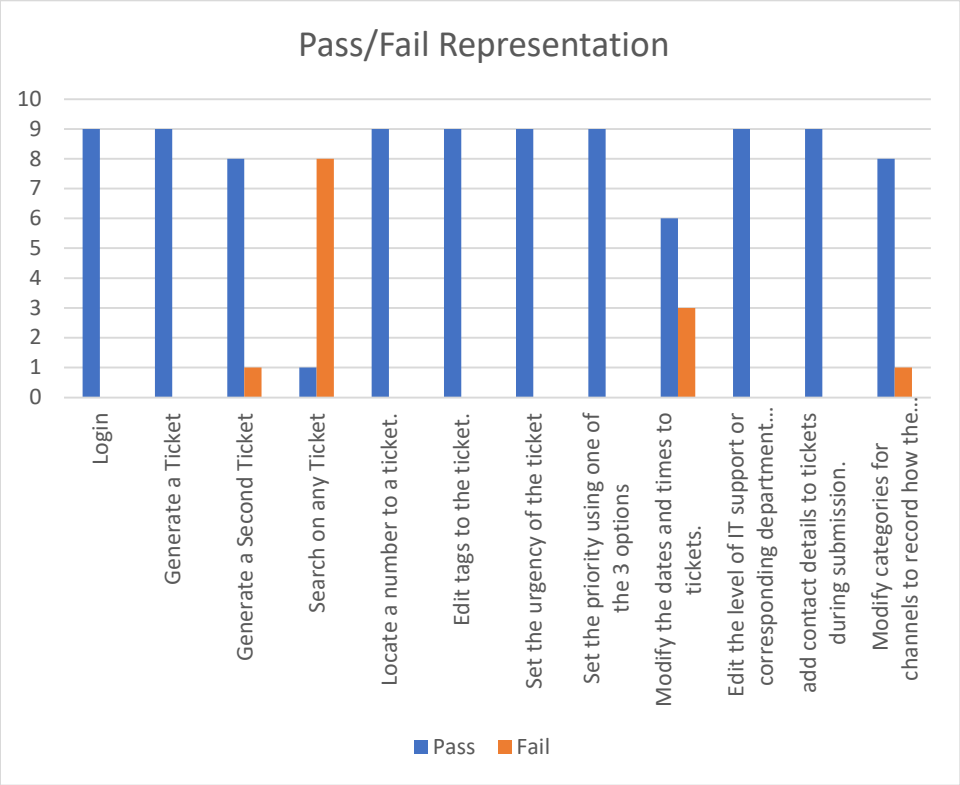
Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	
Generate a Ticket	Medium	10/11/2021	Pass	
Generate a Second Ticket	Complex	10/11/2021	Pass	
Search on any Ticket	Complex	10/11/2021	Fail	Tried to search but didn't seem to produce anything after pressing the search
Locate a number to a ticket.	Medium	10/11/2021	Pass	
Edit tags to the ticket.	Medium	10/11/2021	Pass	
Set the urgency of the ticket	Complex	10/11/2021	Pass	
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	
Modify the dates and times for tickets.	Medium	10/11/2021	Pass/Fail	Time and Date worked roughly as changing the inputs was difficult
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Pass	
add contact details to tickets during submission.	Medium	10/11/2021	Pass	
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	Wasn't sure what channels were used for?

Tester: Mark

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	**Pass**	
Generate a Ticket	Medium	10/11/2021	**Pass**	
Generate a Second Ticket	Complex	10/11/2021	**Pass**	
Search on any Ticket	Complex	10/11/2021	**Fail**	Search was unresponsive
Locate a number to a ticket.	Medium	10/11/2021	**Pass**	
Edit tags to the ticket.	Medium	10/11/2021	**Pass**	
Set the urgency of the ticket	Complex	10/11/2021	**Pass**	
Set the priority using 3 one of the 3 options	Complex	10/11/2021	**Pass**	
Modify the dates and times for tickets.	Medium	10/11/2021	**Pass/Fail**	Couldn't see the drop-down menu for the calendar but I knew about it previously
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	**Pass**	
add contact details to tickets during submission.	Medium	10/11/2021	**Pass**	
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	**Pass**	Could use a tooltip

Tester: Farhan

Scenarios	Complexity	Date of Execution	Status(Pass/Fail/blocked/not executed)	Final Comments
Login	Medium	10/11/2021	Pass	
Generate a Ticket	Medium	10/11/2021	Pass	
Generate a Second Ticket	Complex	10/11/2021	Pass	
Search on any Ticket	Complex	10/11/2021	Pass	
Locate a number to a ticket.	Medium	10/11/2021	Pass	
Edit tags to the ticket.	Medium	10/11/2021	Pass	
Set the urgency of the ticket	Complex	10/11/2021	Pass	
Set the priority using 3 one of the 3 options	Complex	10/11/2021	Pass	
Modify the dates and times for tickets.	Medium	10/11/2021	Pass	
Edit the level of IT support or corresponding department hierarchy.	Complex	10/11/2021	Pass	
add contact details to tickets during submission.	Medium	10/11/2021	Pass	
Modify categories for channels to record how the ticket was submitted.	Complex	10/11/2021	Pass	



The pass-fail data is compiled in the previous chart and suggests a 95% pass rate, one key indicator that can be seen is the Search feature failure. These were the first issues to be fixed as a result of the Beta testing.

Modify date and time seemed to fail in a few tests, upon further investigation we found that the drop-down on the combo box was not clear and the addition of clearer down arrows was added. On this occasion, the fault was to do with the UI more than the function.

Other minor failures such as generating a second ticket and modifying categories were working fine and were more to do with the tester understanding of the task given.

Acceptance Testing.

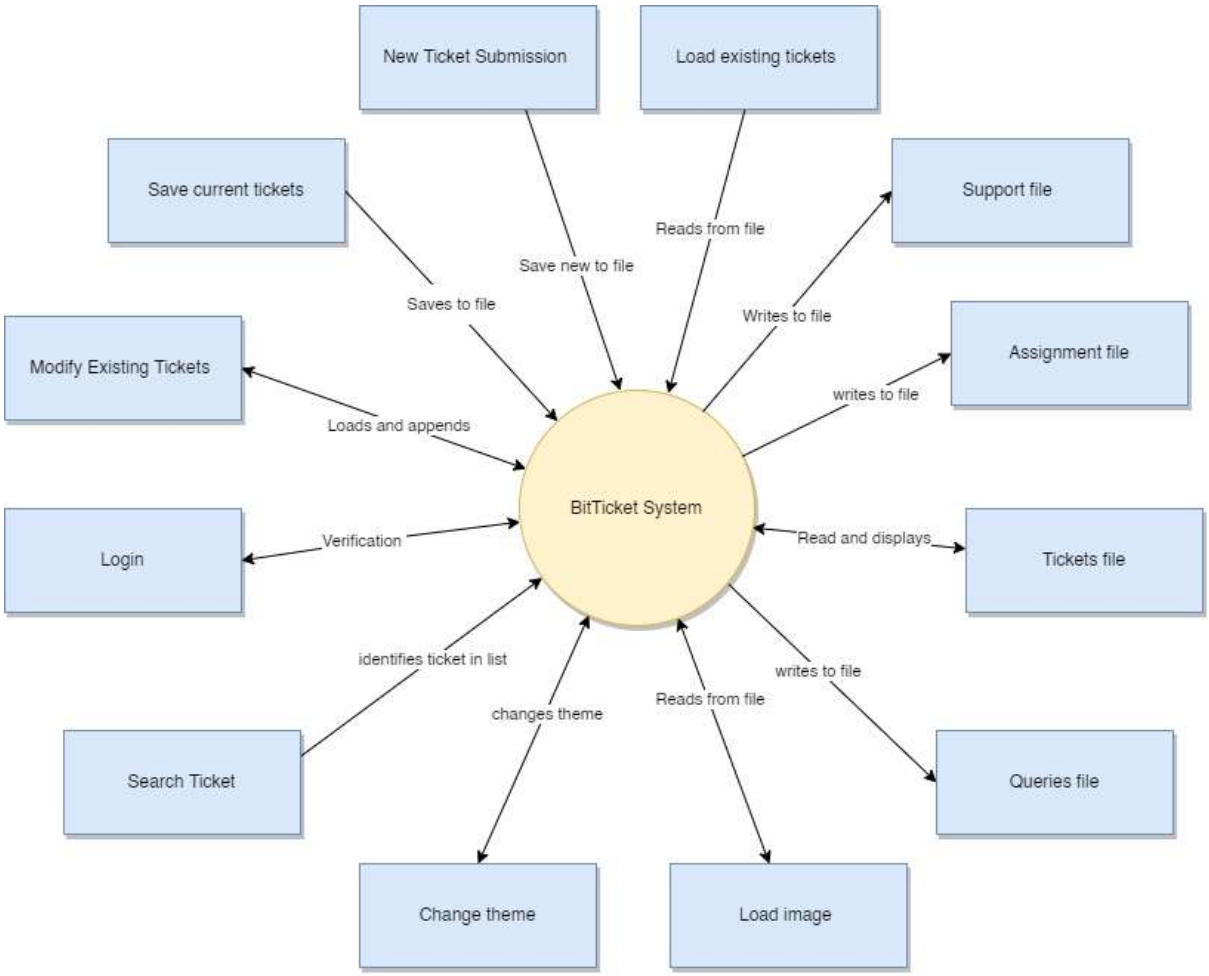
Acceptance Test Report

Test Report ID:	TestMe01		
Product ID / Name:	BitTicket		
Product Version or Build:	1.00		
Present Owner:	Alex Hughes & Mark Pepere		
Created On:	19/11/2021		
Review On:	19/11/2021		
Review By:	Farhan		
Review Comments:	Describe the Comments if any, whether comments have been incorporated or not. If some of the reviewers have minor disagreements, they may note their views before signing off on the document.		
Current Version:	1.00		
Signing Off Authority:	Name Farhan	Position Yoobee Client	Signature, Date Farhan 19/11/2021

http://www.softwaretestinggenius.com

System Documentation.

Context Diagram



Appendix

Administrator: He is the person who has the whole authority over the system.

Developer: He is the person who is responsible to solve the raised bugs.

Tester: He is the person who is responsible to check the system and raise the bugs as encountered.

Manager: A manager is a person whose role is to supervise the developers and testers under him/her.

Defect: A defect can be defined as an anomaly that can disrupt the normal functionality of the system.

Tracking: Tracking means tracing down the life cycle of the bug through its resolution cycle [4].

Software Reliability: Reliability of software means how immune is the software against crashes and how well is the data preserved in case of one.

UML: Unified modelling language

Subsystem: It is the part of the system that performs some specific function.

Component: Component is again a part of the system which provides some specific functionality.

Class: Class is a group of data and embedded functions 60

Module: The module is part of the subsystem which interacts together to make the system work.

System Request: Request made to the server via the system.

Database: The collection of all the data in a system

Roles: Role defines the access rights in a system. Each role has a specific set of rights assigned to it.

Security: Security authorizes users in logging into the system.

References

https://www.zendesk.com

https://www.solarwinds.com

https://www.qt.io

Online Defect Tracking System by Gaurav Soni documentation

https://www.freeprojectz.com/uml-diagram/helpdesk-ticketing-system-uml-diagram

https://qss-stock.devsecstudio.com/

https://www.softwaretestinghelp.com/user-story-acceptance-criteria/