

2207-BSE

# Implementation – Evidence Document

CS106.1

*Alexander Craig, Oliver Anders Grönkrans, Alexander Legner, Liam Konise*

## Document Outline

### Table of Contents

Title Page.....	1
Document Outline.....	2
Table of Contents.....	2
Table of Figures.....	2
Table of Tables .....	3
Changes.....	5
HiFi .....	5
System Architecture.....	6
Activity Diagram.....	6
Class Diagram.....	7
Justification .....	8
Key Functionality.....	9
Function Testing.....	16
Black Box Testing .....	16
User Documentation.....	26
Installation Guide.....	26
User Guide .....	27

### Table of Figures

Figure 1: Hifi Changes .....	5
Figure 2: Activity Diagram.....	6
Figure 3: Class Diagram.....	7
Figure 4: Key Functionality #1.....	9
Figure 5: Key Functionality #2.....	9
Figure 6: Key Functionality #3.....	10
Figure 7: Key Functionality #4.....	11
Figure 8: Key Functionality #5.....	11
Figure 9: Key Functionality #6.....	12
Figure 10: Key Functionality #7.....	13
Figure 11: Key Functionality #8.....	13
Figure 12: Key Functionality #9.....	14
Figure 13: Key Functionality #10.....	15
Figure 14: Functional Testing #1 .....	16
Figure 15: Functional Testing #2 .....	17
Figure 16: Functional Testing #3 .....	18

## Implementation

Figure 17: Functional Testing #4 .....	19
Figure 18: Functional Testing #5 .....	20
Figure 19: Functional Testing #6 .....	21
Figure 20: Functional Testing #7 .....	22
Figure 21: Functional Testing #8 .....	22
Figure 22: Functional Testing #9 .....	23
Figure 23: Functional Testing #10 .....	24
Figure 24: Functional Testing #11 .....	25
Figure 25: Installation guide .....	26
Figure 26: User Guide #1.....	27
Figure 27: User Guide #2.....	28
Figure 28: User Guide #3.....	29
Figure 29: User Guide #4.....	30
Figure 30: User Guide #5.....	31
Figure 31: User Guide #6.....	32
Figure 32: User Guide #7.....	33

## Table of Tables

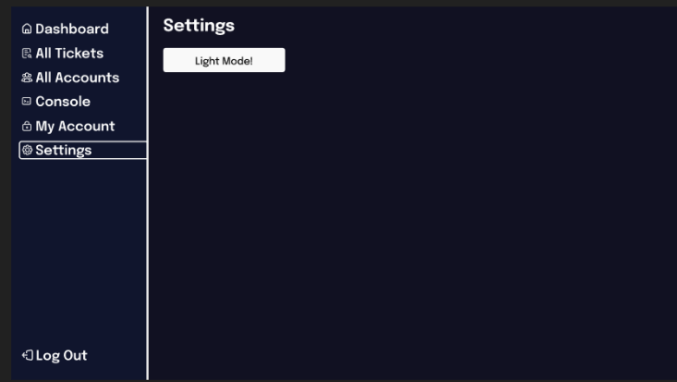
No table of figures entries found.

Intentionally Blank

## Changes

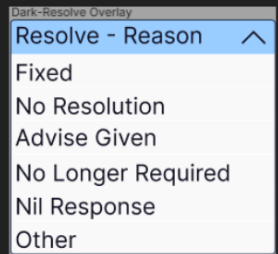
### HiFi

#### Changes



#### Settings/Lightmode

- Removed completely as the feature Lightmode has been deemed not important to the target audience
- All other data that could be included in settings is found elsewhere



#### Resolve Ticket popup

- The resolve ticket drop down has been replaced to a pop up to better accentuate its importance

Figure 1: Hifi Changes

## System Architecture

### Activity Diagram

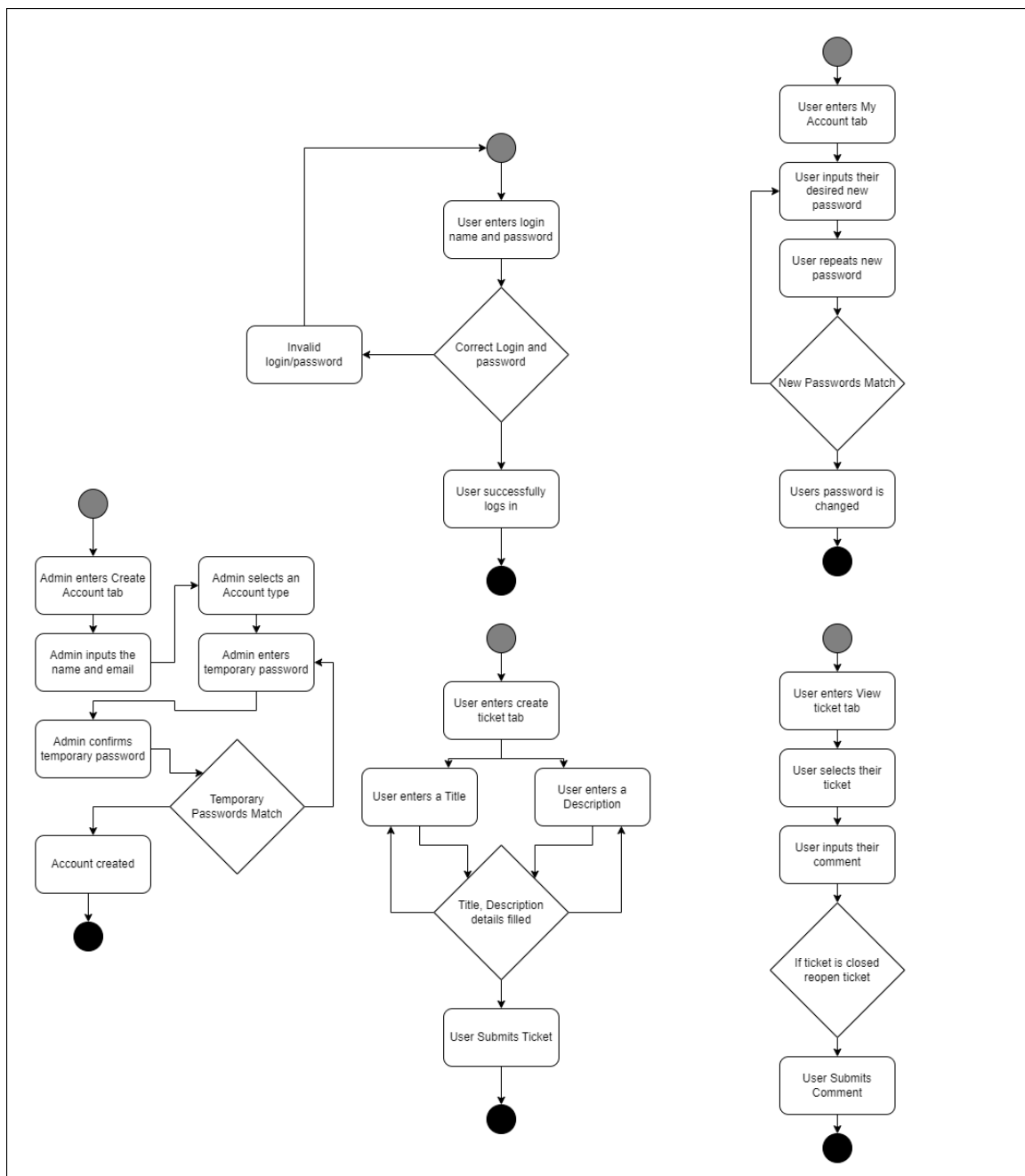
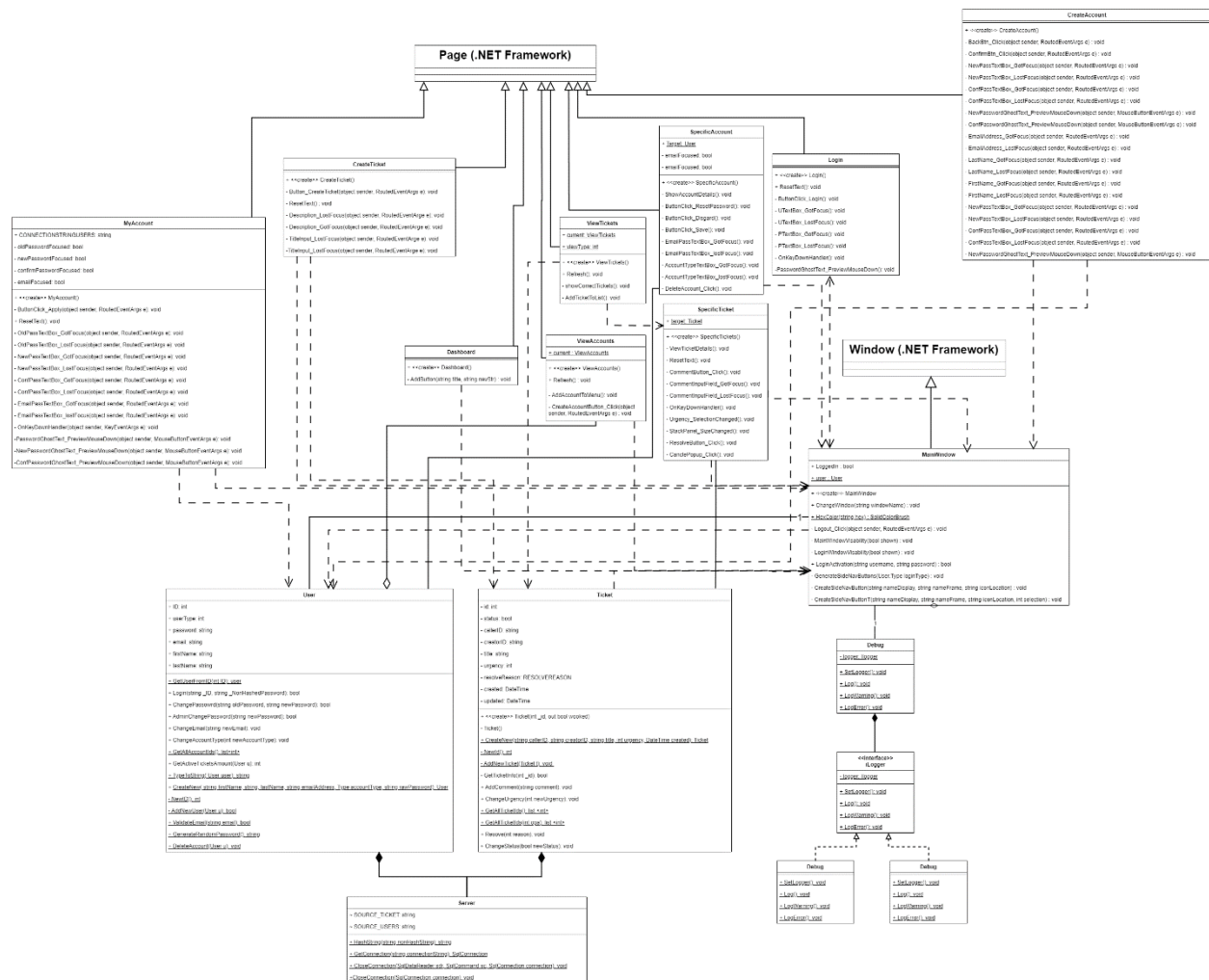


Figure 2: Activity Diagram

## Class Diagram



## Implementation

### Justification

The software is built on a WPF/.NET frame, in which instances of objects such as tickets and users are stored in local variables, but these load variables from a SQL server on the local machine by querying the corresponding keywords.

A user instance is created by querying the id of the user (and in cases such as login comparing password as well), loading the fields into the corresponding variable, such as FirstName goes into firstName in the application.

The main deviation from this structure is comments, which are stored with a limit character between each individual comment of a ticket, and an example of a stored comment thread would be:

◆User|One|2023/05/20-19:10|Heres My Comment string◆User|Two|2023/05/20-20:19|This is another comment

Passwords are hashed before being sent to the SLQ database, and thus all login attempts hashes the users input password, against the stored password hashes. For ticket indexing, checking if a ticket belongs to a user, it simply compares caller and creator ID's against the logged in user's ID.

The databases are divided into two SQL servers, Tickets.mdf (which contains the table AllTickets), and Users.mdf (which contains the table Users). The user database is queried when handling logins, account creation, updating names, emails, passwords, etc, while the ticket database is queried when adding or editing tickets.



## Implementation

### Key Functionality

#### Project Functionality Screenshots:

##### Logging in:

When the “LogIn” button is clicked in the system “ButtonClick\_Login” runs and checks if the password and username is correct and if not the “MessageBox Result” = “Incorrect Credentials”.

C#:

```
/// <summary>
/// trys to login the user with credentials from the user
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void ButtonClick_Login(object sender, RoutedEventArgs e)
{
    // log the user in - if unsuccessful alert user and reset textboxes
    if(!((MainWindow)Application.Current.MainWindow).LoginActivation(LoginUserName.Text, LoginPassword.Password))
    {
        ResetText();
        MessageBoxResult wrongCredentials = MessageBox.Show("Incorrect credentials!");
    }
}
```

Figure 4: Key Functionality #1

##### System:

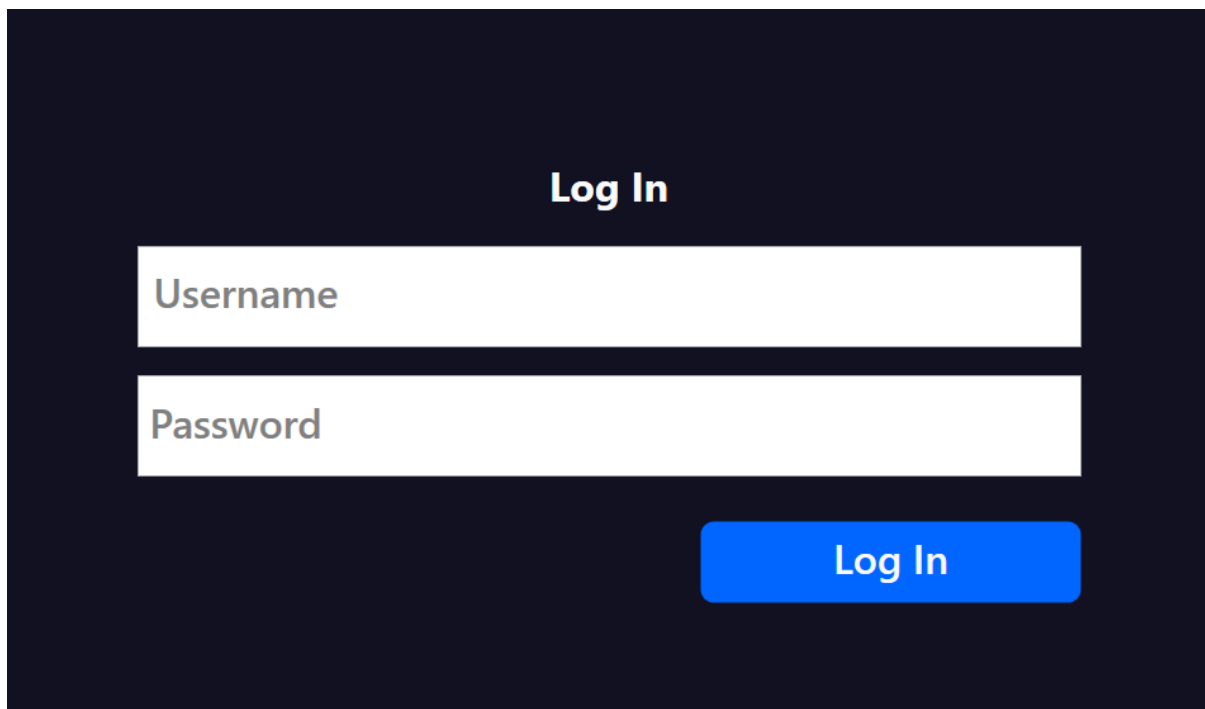
The screenshot shows a dark-themed login window titled "Log In". It features two white input fields: the top one is labeled "Username" and the bottom one is labeled "Password". Below these fields is a blue button with the text "Log In" in white. The entire interface is centered on a dark background.

Figure 5: Key Functionality #2

## Implementation

### Create Ticket:

When Submit is Selected, System checks Title ("TitleInput.text") Urgency (Urgency.SelectedIndex") Creator ID ("current.ID.ToString()") who its created for ("CreatedFor.text") and lastly the description ("Description.Text") and saves the data to the database. If Description and Title are not filled in Users will be shown a text box via the "if" statement. The ticket is then created and the user's view is now replaced by the ticket they just created.

C#:

```
public CreateTicket()
{
    User current = MainWindow.user;
    InitializeComponent();
    CreatedBy.Text = current.ID.ToString(); // sets created by to this user
    CreatedFor.Text = current.ID.ToString(); // sets created for to this user by default (can be changed while in application)
}

/// <summary>
/// creates a ticket if required fields are valid
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void Button_CreateTicket(object sender, RoutedEventArgs e)
{
    User current = MainWindow.user; // get current user logged in

    // get all variables from XAML inputs
    string title = TitleInput.Text;
    int urgency = Urgency.SelectedIndex + 1; // 1 2 3 for high medium low
    string creatorID = current.ID.ToString();
    string createdFor = CreatedFor.Text;
    string description = Description.Text;

    // check all required values are valid (stop if invalid)
    if (TitleInput.Text == "Title" || TitleInput.Text == "") // IF THE USER HAS NOT ENTERED A TITLE
    {
        MessageBox.Show("Please enter a title");
        return;
    }
    else if (Description.Text == "Description" || Description.Text == "") // IF THE USER HAS NOT ENTERED A DESCRIPTION
    {
        MessageBox.Show("Please enter a description");
        return;
    }

    // create ticket
    Ticket t = Ticket.CreateNew(createdFor, creatorID, title, urgency, DateTime.Now);
    t.AddComment(description);

    // change the window to view the newly created ticket
    MainWindow window = (MainWindow)Application.Current.MainWindow;
    SpecificTicket.target = t;
    window.ChangeWindow("SpecificTicket.xaml");
}
```

Figure 6: Key Functionality #3

System:

## Implementation

The screenshot shows a web application interface with a dark blue sidebar on the left and a main content area on the right. The sidebar contains a 'Dashboard' link at the top, followed by 'All Tickets', 'All Accounts', 'Create Ticket' (highlighted), 'Create Account', 'My Account', 'Settings', and 'Log Out' at the bottom. The main content area is titled 'Create Ticket' and contains a form with the following fields: 'Title' (text input), 'Urgency' (dropdown menu with 'Low' selected), 'Created For' (text input with '7'), 'Created By' (text input with '7'), and 'Description of problem' (text area with 'Description'). A blue 'Submit' button is located at the bottom right of the form.

Figure 7: Key Functionality #4

### Creating Account:

When creating an account and confirm is selected the system checks for Name (“firstName.Text”, “lastName.Text”) Email (“EmailAddress.Text”) Account Type (“AccountType.SelectedIndex”) and Password (“NewPassword.Password”). If these details are not filled in, the “else if” statement will trigger the respective “MessageBox.Show” to trigger.

C#:

```
/// <summary>
/// creates the account if the data is valid
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void ConfirmBtn_Click(object sender, RoutedEventArgs e)
{
    if(NewPassword.Password == ConfPassword.Password && User.ValidateEmail(EmailAddress.Text))
    {
        User u = User.CreateNew(FirstName.Text, LastName.Text, EmailAddress.Text, (User.Type)AccountType.SelectedIndex + 1, NewPassword.Password);
        SpecifcAccount.target = u;
        MainWindow mw = (MainWindow)Application.Current.MainWindow;
        mw.ChangeWindow("SpecifcAccount.xaml");
    }
    else if (!User.ValidateEmail(EmailAddress.Text))
    {
        MessageBoxResult invalidEmail = MessageBox.Show("Email address already in use!");
    }
}
```

Figure 8: Key Functionality #5

System:

**Create Account**

**Name**

First Name

Last Name

**E-mail address**

Enter Email Address

**Account Type**

1 - User

**Password**

New password

Confirm new password

**This is a temporary password!**

Back Confirm

Log Out

Figure 9: Key Functionality #6

#### Add Comment:

String “amended comment” allows user to input a comment in a ticket while saving and inserting their Name and Time. When the button “Submit Comment” is clicked the input comment will “Try” to add the comment to the ticket in the database and if it fails it alerts the user (“Catch”).

C#:

## Implementation

```
public void AddComment(string comment)
{
    try
    {
        string amendedComment = "" + MainWindow.user.firstName + "!" + MainWindow.user.lastName + "!" + DateTime.Now.ToString() + "!" + comment;
        comments.Add(amendedComment);
        amendedComment = string.Empty;
        foreach (string c in comments)
        {
            amendedComment += c + ' ';
        }
        if (amendedComment.EndsWith(" "))
        {
            amendedComment = amendedComment.Remove(amendedComment.Length - 1, 1); // remove last symbol
        }

        using (SqlConnection connection = Server.GetConnection(Server.SOURCE_TICKET))
        {
            // FILESTREAM / WRITER, ALLOWS INSERTING / UPDATING ROWS IN SQL
            SqlDataAdapter adapter = new SqlDataAdapter();
            string commandText = "UPDATE AllTickets SET COMMENTS=@comment WHERE ID='" + this.id + "'";
            adapter.InsertCommand = new SqlCommand(commandText, connection);
            adapter.InsertCommand.Parameters.AddWithValue("@comment", amendedComment);
            adapter.InsertCommand.ExecuteNonQuery();
            Server.CloseConnection(connection);
        }

        using (SqlConnection connection = Server.GetConnection(Server.SOURCE_TICKET))
        {
            SqlDataAdapter adapter = new SqlDataAdapter();
            string commandText = "UPDATE AllTickets SET UPDATED='" + DateTime.Now.ToString() + "' WHERE ID='" + this.id + "'";
            adapter.InsertCommand = new SqlCommand(commandText, connection);
            adapter.InsertCommand.ExecuteNonQuery();
            Server.CloseConnection(connection);
        }
    }
    catch (Exception e)
    {
        Debug.LogWarning("Operation Unsuccessful - " + e.Message);
        MessageBox.Show("Operation was not successful!\nPlease try again...", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Figure 10: Key Functionality #7

System:

Dashboard

All Tickets

All Accounts

Create Ticket

Create Account

My Account

Settings

Log Out

Ticket #	Title	Status	Created	Updated
INC1	1	Closed	1d ago	1d ago

Caller

Created by

Urgency

1

1

Low

TA

OA

OA

OA

The Admin

RESOLVED TICKET WITH REASON Nil Response.

1d ago

Oliver Anders Grönkrans

why the fuck is this like this

1d ago

Oliver Anders Grönkrans

dfhdfhrehrerheu

1d ago

Oliver Anders Grönkrans

dwefwefwef

1d ago

Figure 11: Key Functionality #8

## Implementation

### Change password:

When changing password the system checks the database to see if your old password matches (“oldPassword = Server.HashString(oldPassword)”) and if your new password matches the confirmed password changes (“newPassword = Server.HashString(newPassword)”) if not the user is alerted (“Catch”).

C#:

```
/// <summary>
/// changes the password of the current instance of the user if the old password matches the current password
/// </summary>
/// <param name="oldPassword"></param>
/// <param name="newPassword"></param>
/// <returns></returns>
1 reference
public bool ChangePassword(string oldPassword, string newPassword)
{
    try
    {
        oldPassword = Server.HashString(oldPassword);
        newPassword = Server.HashString(newPassword);
        // CHECKS IF THE NEW PASSWORDS MATCHES, AND IF THE OLD PASSWORD MATCHES THEIR CURRENT PASSWORD
        if (oldPassword == password)
        {
            SqlConnection connection = Server.GetConnection(Server.SOURCE_USERS);
            SqlDataAdapter adapter = new SqlDataAdapter();

            string commandText = "UPDATE Users SET Password=@password WHERE ID='" + ID + "'";
            adapter.InsertCommand = new SqlCommand(commandText, connection);
            adapter.InsertCommand.Parameters.AddWithValue("@password", newPassword);
            adapter.InsertCommand.ExecuteNonQuery();

            Server.CloseConnection(connection);
            password = newPassword;
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (Exception e)
    {
        Debug.LogWarning("Operation Unsuccessful - " + e.Message);
        MessageBox.Show("Operation was not successful!\nPlease try again...", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
        return false;
    }
}
```

Figure 12: Key Functionality #9

## Implementation

System:

The screenshot displays a web application interface for account management. On the left is a dark sidebar with white text and icons for navigation: Dashboard (house icon), All Tickets (list icon), All Accounts (people icon), Create Ticket (plus icon), Create Account (plus icon with person), My Account (lock icon), and Settings (gear icon). At the bottom of the sidebar is a 'Log Out' button with a back arrow icon. The main content area has a dark background and is titled 'My Account' in large white font. Below the title, there are several form fields: 'Account ID' with a value of '#7', 'Name' with a value of 'The Admin', and 'E-mail address' with a value of 'admin'. A 'Change Password' section contains three password input fields, each masked with six asterisks. A blue 'Apply Changes' button is positioned at the bottom right of the form. A small white modal dialog box is overlaid on the right side of the form, displaying the message 'Successfully updated password!' and an 'OK' button.

Figure 13: Key Functionality #10

## Implementation

### Function Testing

#### Black Box Testing

##### Test cases

1. Input “bad” input (spaces, special characters, etc) in login form

##### Expected output

Login attempt fails as with any other invalid credentials.

##### Used parameter

Username, password: 0 1 2 3 4 5 6 7 8 9 ! “ # ¤ % & / ( ) = ? ` | < >

##### Received output

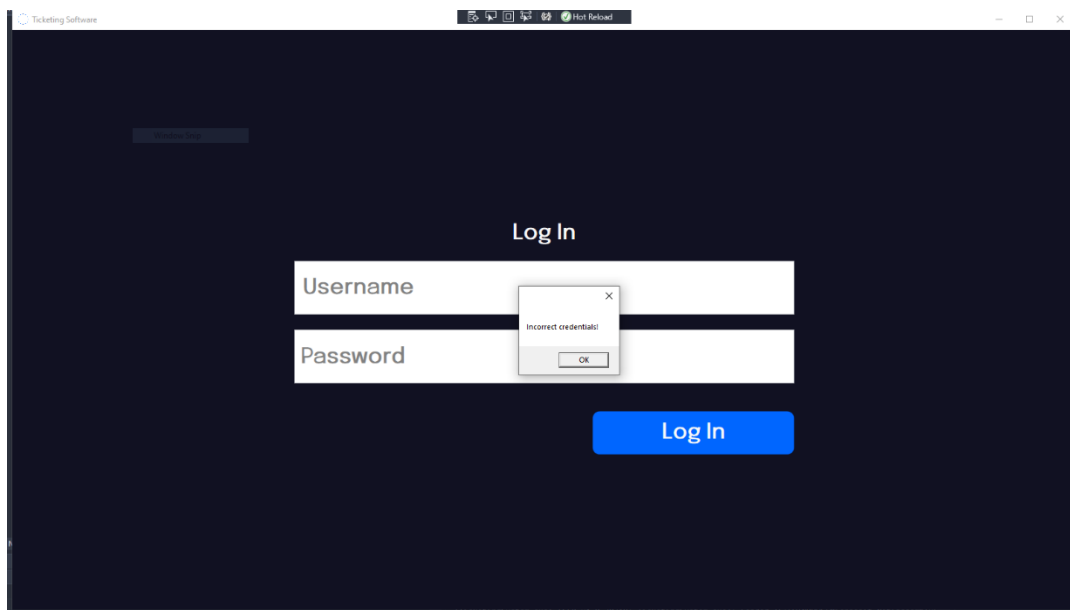


Figure 14: Functional Testing #1

##### Result

Passed

2. Input spaces and special characters in new ticket

##### Expected output

Ticket is created with corresponding data without issue, and the same data can be loaded in the ticket view page.



## Implementation

### Used parameters

Title, Caller, Description: 0 1 2 3 4 5 6 7 8 9 ! " # \$ % & / ( ) = ? ` | < >

### Received output

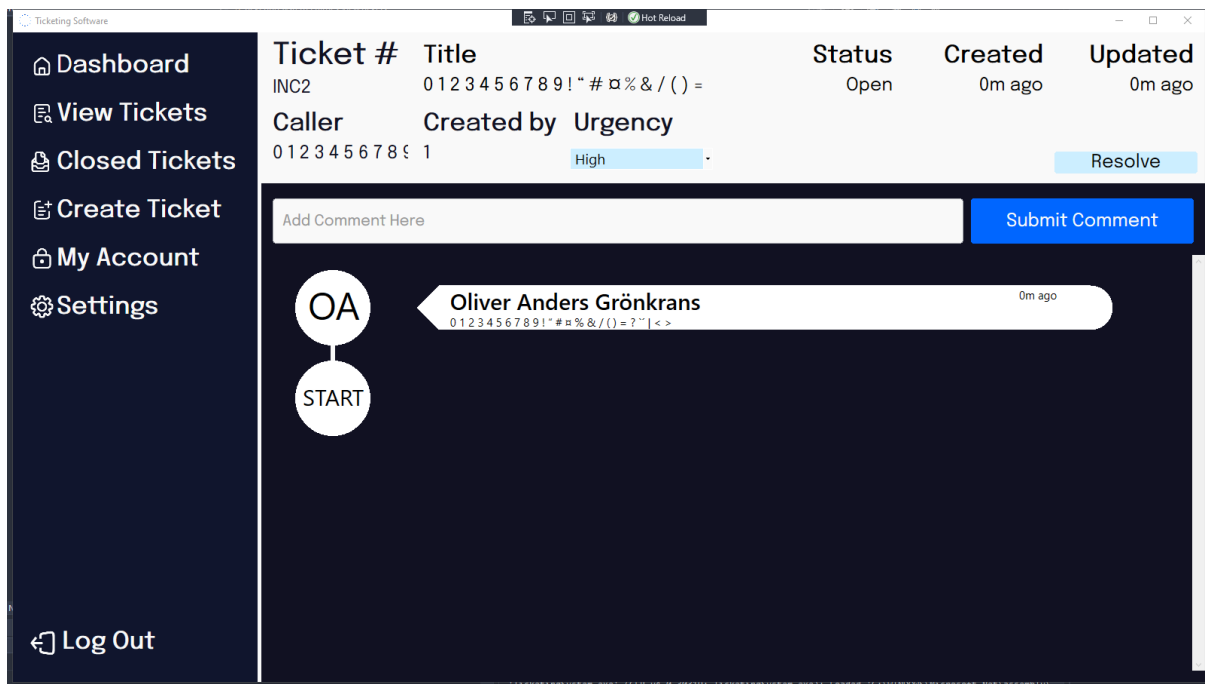


Figure 15: Functional Testing #2

### Result

Passed

### 3. Resolve ticket

#### Expected output

Changes documented in comment field without issue.

### Used parameters

Resolve status - Fixed

### Received output

## Implementation

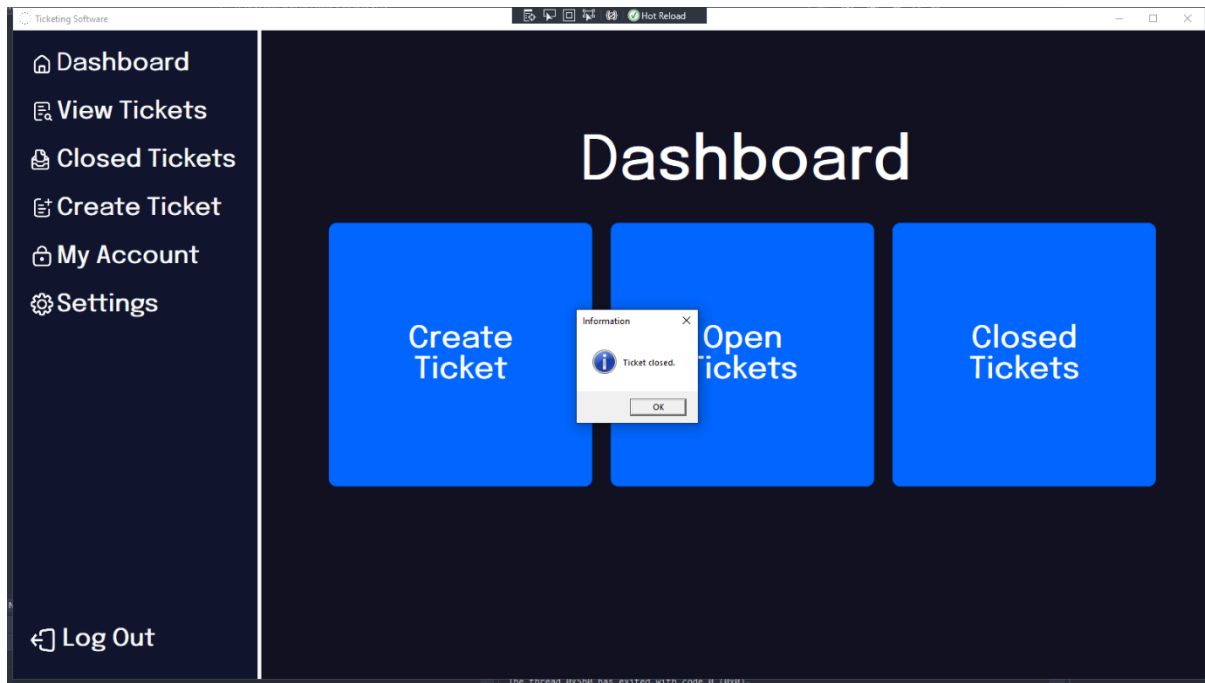


Figure 16: Functional Testing #3

## Result

Passed

### 4. Reopen ticket with comment containing spaces and special characters

#### Expected output

Ticket is reopened, comment is added, and status update is added without issue.

#### Used parameters

Comment text: 0 1 2 3 4 5 6 7 8 9 ! " # \$ % & / ( ) = ? ` | < >

#### Received output

## Implementation

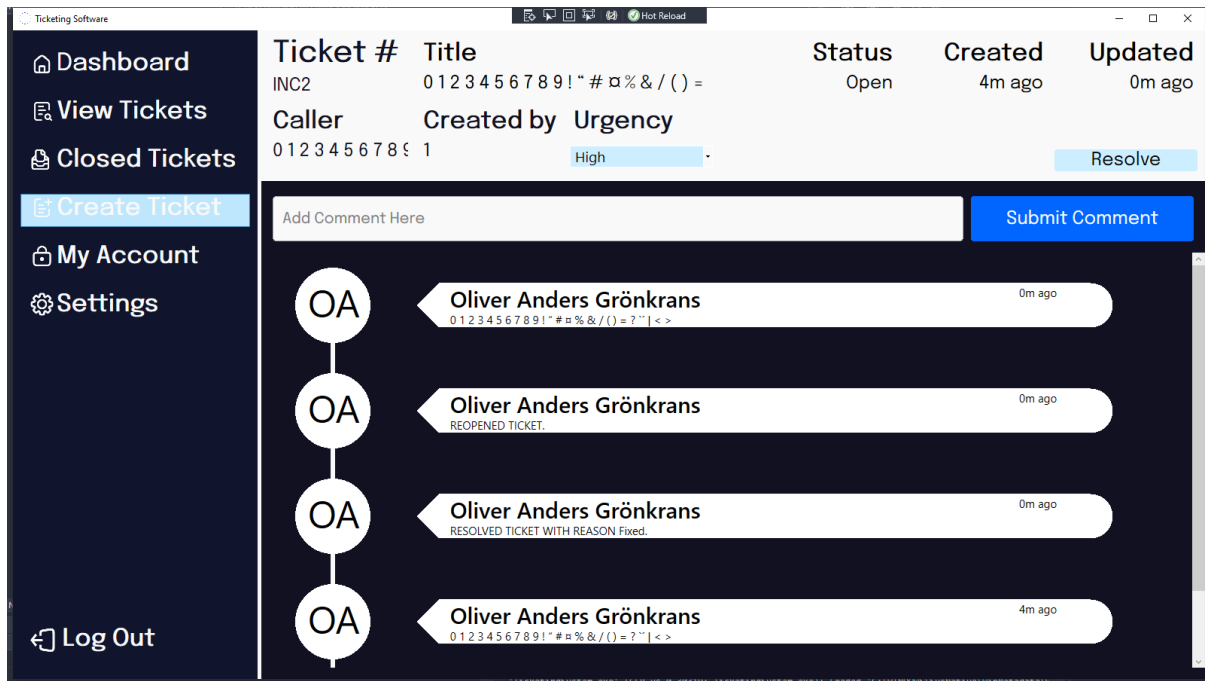


Figure 17: Functional Testing #4

### Result

Passed

#### 5. Try to create account with an e-mail address which is already in use

#### Expected output

Account creation is denied, with error message stating that the e-mail address already is in use.

#### Used parameters

Email: [270045020@yoobeestudent.ac.nz](mailto:270045020@yoobeestudent.ac.nz) (used by user 1)

#### Received output

## Implementation

The screenshot shows a web application interface for creating a new account. The left sidebar contains navigation links: Dashboard, All Tickets, All Accounts, Create Ticket, Create Account, My Account, and Settings. A 'Log Out' button is at the bottom of the sidebar. The main content area is titled 'Create Account' and contains the following form fields:

- Name:** A text input field with the value 'New'.
- User:** A text input field with the value 'User'.
- E-mail address:** A text input field with the value '270045020@yoobeestudent.'. A modal dialog is open over this field, displaying the message 'Email address already in use!' with an 'OK' button.
- Account Type:** A dropdown menu with the value '1 - User'.
- Password:** Two text input fields, both containing three asterisks (\*\*\*). A red warning message 'This is a temporary password!' is displayed next to the second password field.

At the bottom of the form are two buttons: 'Back' (light blue) and 'Confirm' (blue).

Figure 18: Functional Testing #5

### Result

Passed

#### 6. Create account with two first names and two last names

##### Expected output

Account is created without issue and the name is displayed correctly.

##### Used parameters

First name: Fredrik Anders

Last name: Andersson Stigstorp

##### Received output

## Implementation

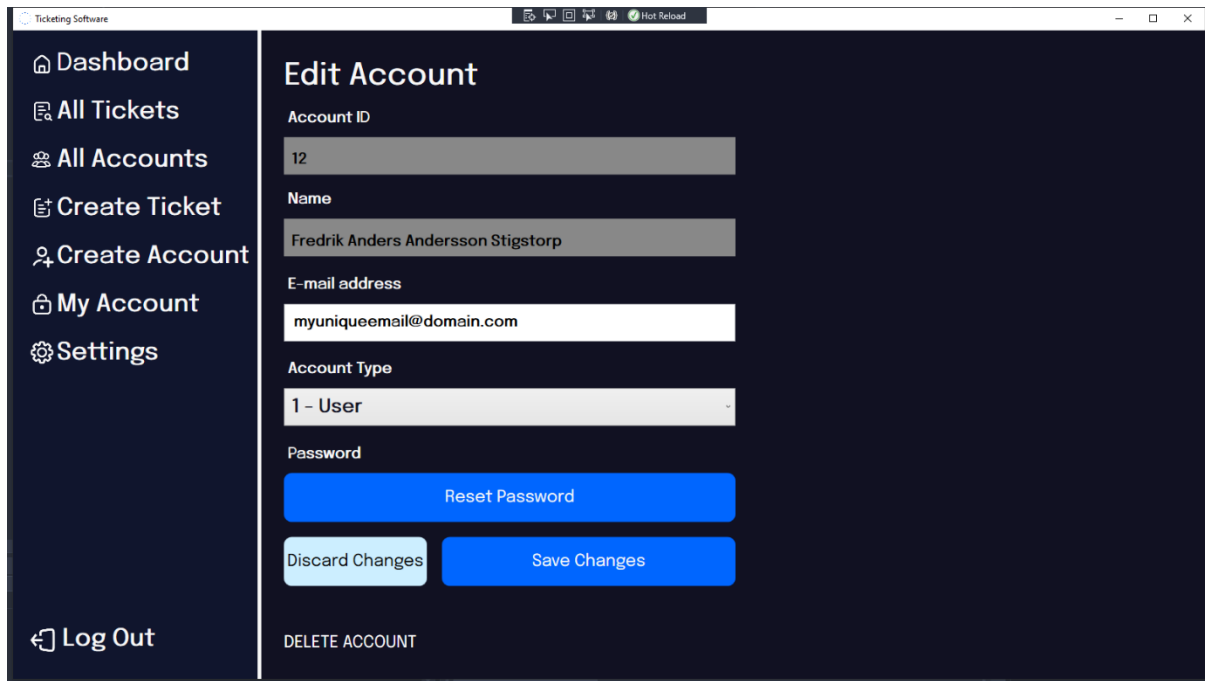


Figure 19: Functional Testing #6

### Result

Passed

#### 7. Delete account with ID 8

##### Expected output

The account with ID 8 (and no other account) is deleted without issue.

##### Used parameter

User: ID 8 out of 10

##### Received outputs

## Implementation

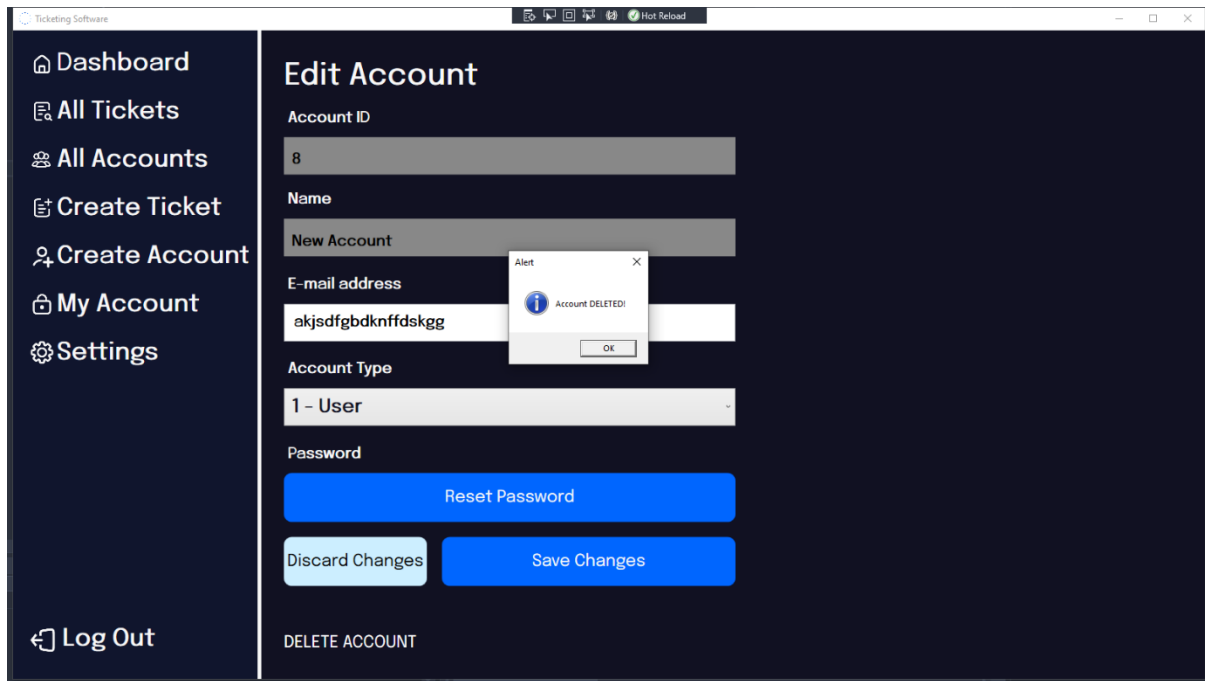


Figure 20: Functional Testing #7

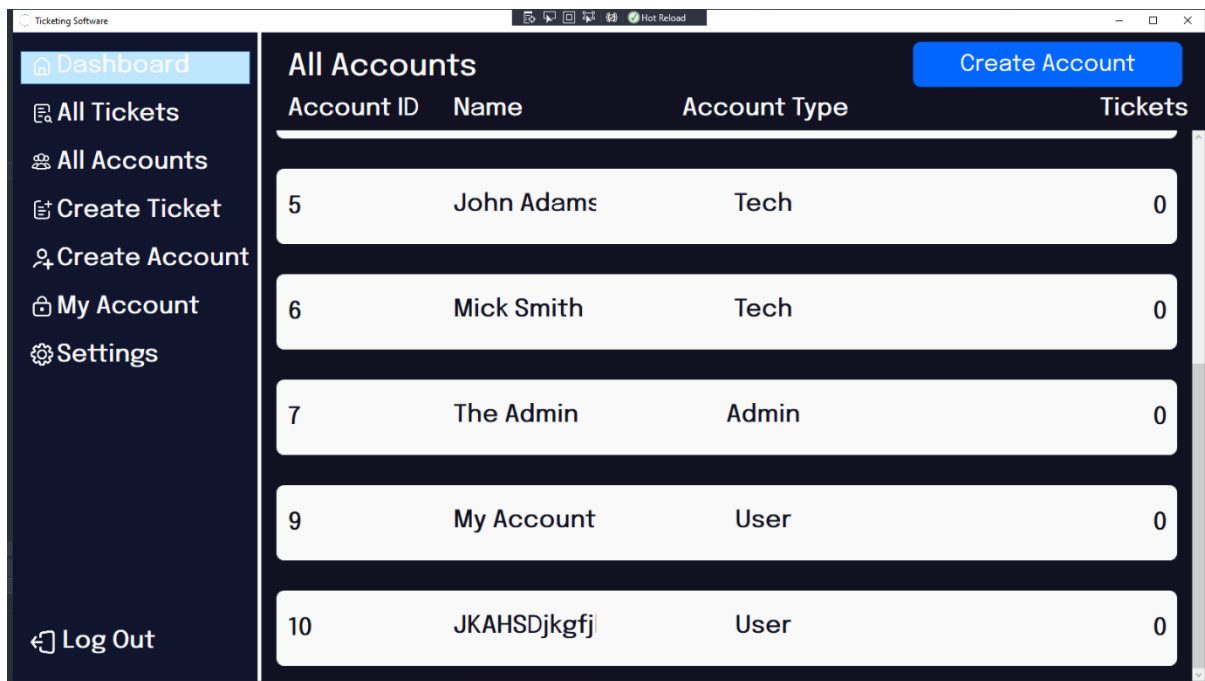


Figure 21: Functional Testing #8

## Result

Passed

## Implementation

### 8. Create new account after deleting account with non-edge ID

#### Expected output

New account gets ID of one higher than the last account in the database, and not an ID generated of the length of the database which would result in overlapping ID's.

#### Used parameters

Database: Contains users 1-7, 9-10

First name: Lisbeth

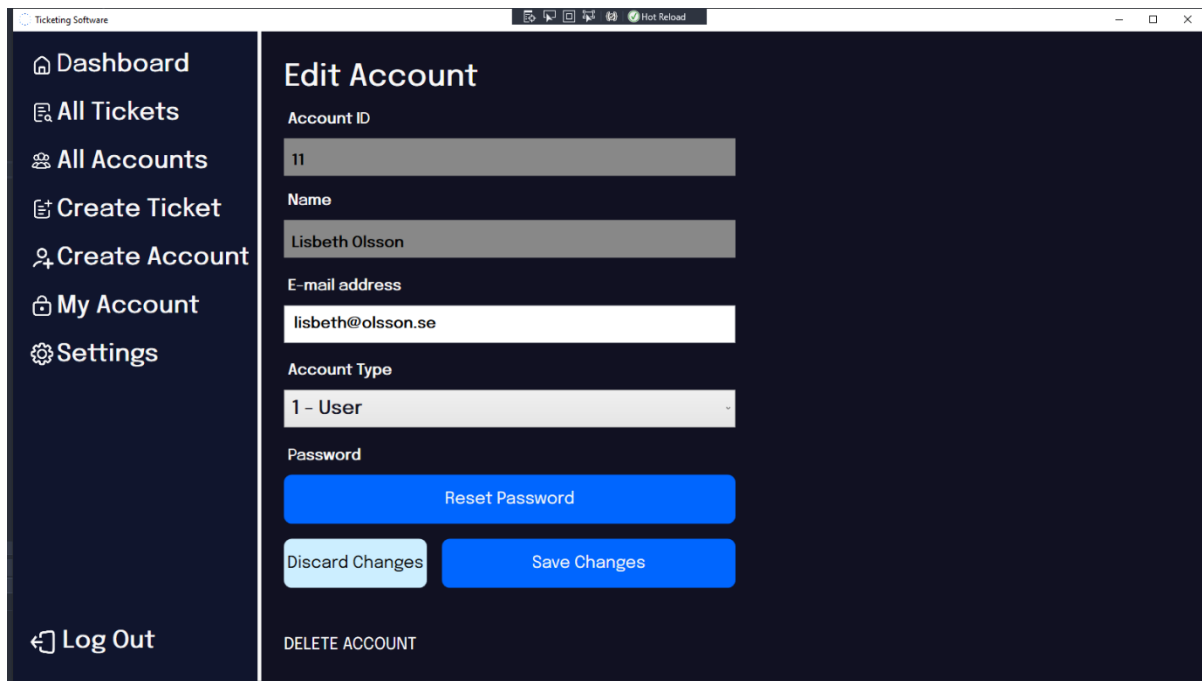
Last name: Olsson

E-mail: [lisbeth@olsson.se](mailto:lisbeth@olsson.se)

Account type: 1 (user)

Password: 123

#### Received output



The screenshot shows a web application window titled 'Ticketing Software'. On the left is a dark sidebar with navigation links: Dashboard, All Tickets, All Accounts, Create Ticket, Create Account, My Account, Settings, and Log Out. The main content area is titled 'Edit Account' and contains the following fields and buttons:

- Account ID:** A text input field containing the value '11'.
- Name:** A text input field containing the value 'Lisbeth Olsson'.
- E-mail address:** A text input field containing the value 'lisbeth@olsson.se'.
- Account Type:** A dropdown menu showing '1 - User'.
- Password:** A section containing a blue 'Reset Password' button.
- At the bottom of the form are two buttons: 'Discard Changes' and 'Save Changes'.
- Below the buttons is a link labeled 'DELETE ACCOUNT'.

Figure 22: Functional Testing #9

#### Result

Passed

## Implementation

### 9. Create ticket for other user (as in technician creates a ticket for a user)

#### Expected output

Ticket is created and the user which it is created for can access it.

#### Used parameters

Creator: User with ID 7

Caller: User with ID 1

Ticket title: Ticket for other user

Ticket description: This should be accessible to user 1

#### Received output

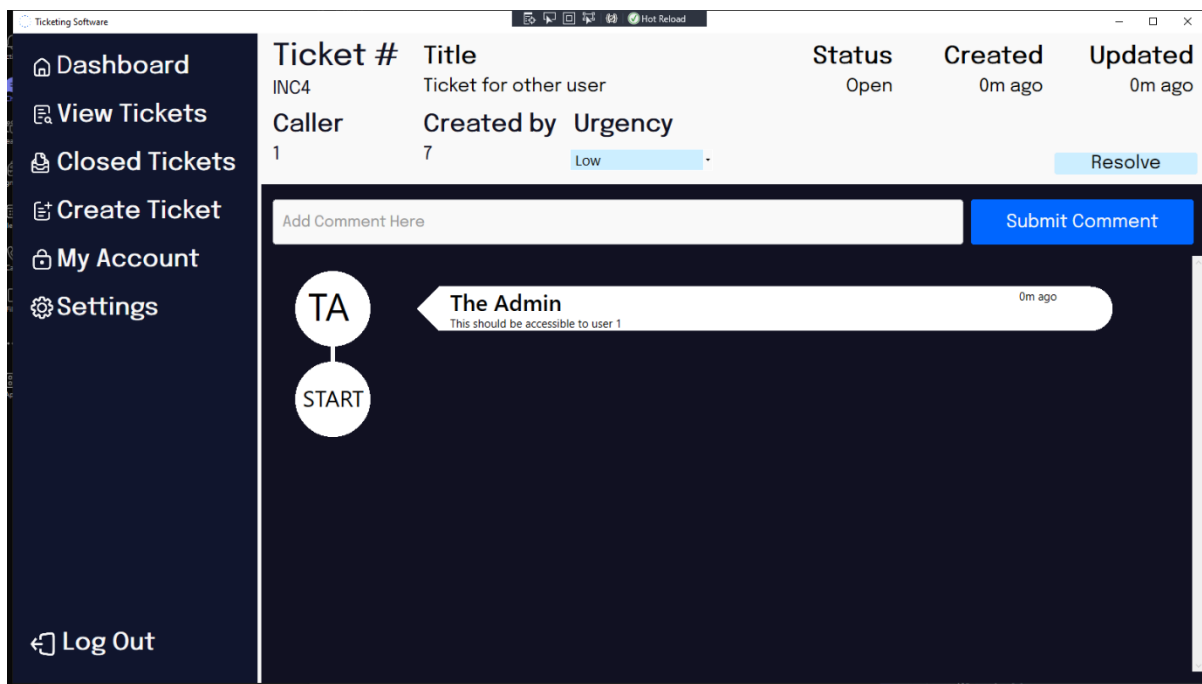


Figure 23: Functional Testing #10

#### Result

Passed

### 10. Add comment as caller in a multi-user ticket

#### Expected output



## Implementation

The created comment is added and displayed correctly, with the username of the user who is adding it.

### User parameters

Ticket: INC4 (Ticket from previous test)

Creator: User with ID 7

Caller/Commenter: User with ID 1

Comment text: This is my own comment

### Received output

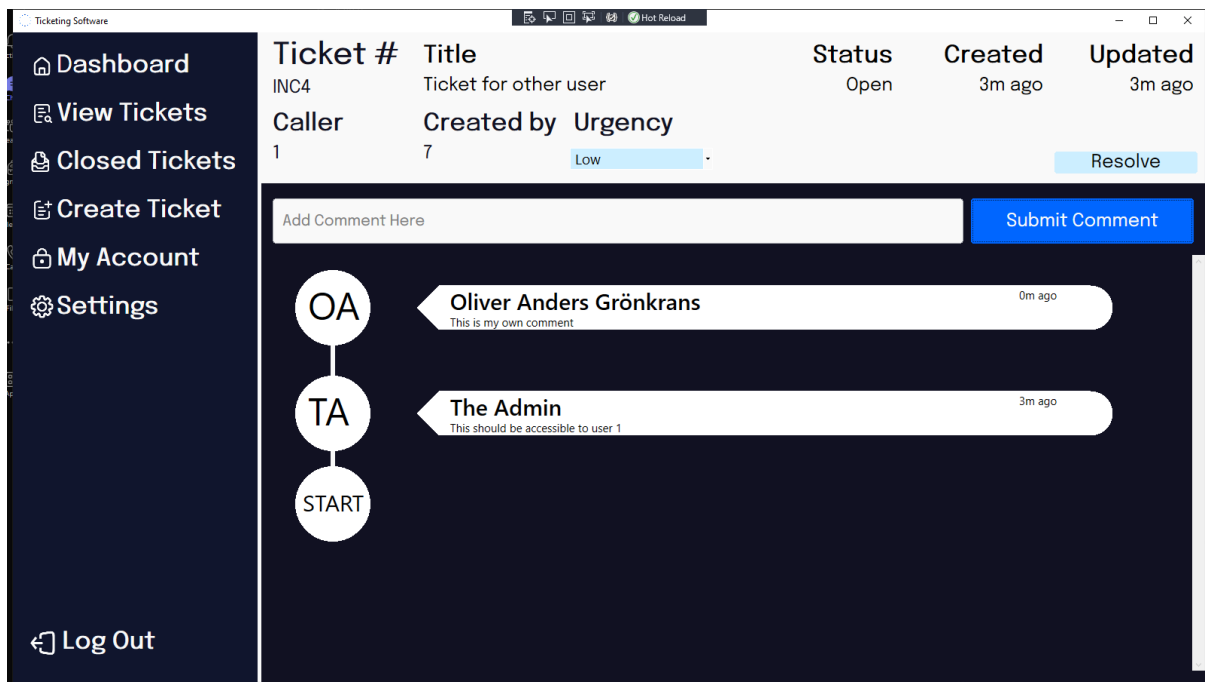


Figure 24: Functional Testing #11

### Result

Passed

## Implementation

## User Documentation

Easier to read version is on the GitHub repo: <https://github.com/ilexl/CS106>

## Installation Guide

### CS106 Ticketing System Guide

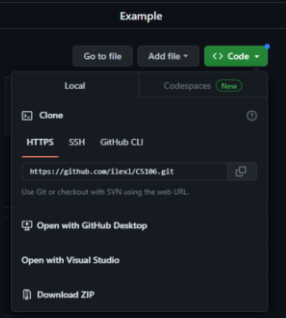
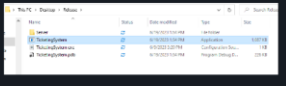

This guide is intended to get any user to be able to setup the ticketing system on their system.

#### Requirements

- Windows 10
- Region/Language Settings (English United States)
- 50 MB (Minimum for installation - increases with use)
- Minimum screen resolution 1600x900

#### Installation/Setup Guide Guide

##### Initial Setup

Instruction	Example
1. Download the repository zip file	
2. Extract the folder from within the zip file Find the folder called "Release" which contains the TicketingSystem.exe and other folders within. Drag this folder where ever you like.	
3. Open the TicketingSystem application	

#### Admin Guide - Setup

- Initial Setup - Log in with the default username: admin | password: admin
- Navigate to 'My Account'
- ⚠ Enter the old password: admin | and create a new password for the admin account: ⚠
- You can then manage all the users and tickets in this application using the 'All Accounts' button on the side nav
- The buttons and instructions are self explanatory when it comes to managing accounts and tickets.
- You will need to create accounts for other admins, technicians and users so they can access the application tool!

Figure 25: Installation guide

# User manual

## Admin Guide - Features

### Create an account

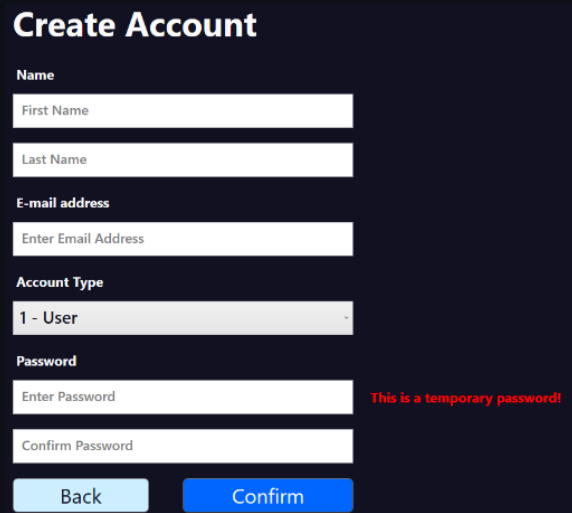
Instruction	Example
<ul style="list-style-type: none"><li>- Navigate to 'create account'</li><li>- Fill out all the new account details</li><li>- ⚠ Make sure you select the correct account type ⚠</li><li>- Enter a temporary password for the new user</li><li>- Give the login details (ID, email, temporary password) to the new user and tell them to change the password</li></ul>	 <p><b>Create Account</b></p> <p><b>Name</b></p> <p>First Name</p> <p>Last Name</p> <p><b>E-mail address</b></p> <p>Enter Email Address</p> <p><b>Account Type</b></p> <p>1 - User</p> <p><b>Password</b></p> <p>Enter Password</p> <p>Confirm Password</p> <p>Back Confirm</p> <p>This is a temporary password!</p>

Figure 26: User Guide #1

[Edit an account](#)

Instruction	Example																								
- Navigate to 'all accounts'	<div><b>All Accounts</b><div>Create Account</div><table><thead><tr><th>Account ID</th><th>Name</th><th>Account Type</th><th>Tickets</th></tr></thead><tbody><tr><td>5</td><td>John Adams</td><td>Tech</td><td>0</td></tr><tr><td>6</td><td>Mick Smith</td><td>Tech</td><td>0</td></tr><tr><td>7</td><td>The Admin</td><td>Admin</td><td>0</td></tr><tr><td>8</td><td>Example Account</td><td>User</td><td>4</td></tr><tr><td>9</td><td>t t</td><td>Error - Unknc</td><td>0</td></tr></tbody></table></div>	Account ID	Name	Account Type	Tickets	5	John Adams	Tech	0	6	Mick Smith	Tech	0	7	The Admin	Admin	0	8	Example Account	User	4	9	t t	Error - Unknc	0
Account ID	Name	Account Type	Tickets																						
5	John Adams	Tech	0																						
6	Mick Smith	Tech	0																						
7	The Admin	Admin	0																						
8	Example Account	User	4																						
9	t t	Error - Unknc	0																						
<div><h2>Edit Account</h2><div>Account ID</div><div>9</div><div>Name</div><div>t t</div><div>E-mail address</div><div>t</div><div>Account Type</div><div>1 - User</div><div>Password</div><div>Reset Password</div><div>Discard Changes</div><div>Save Changes</div></div>																									
- Select an account - Make edits as per sections																									

Figure 27: User Guide #2

## Implementation

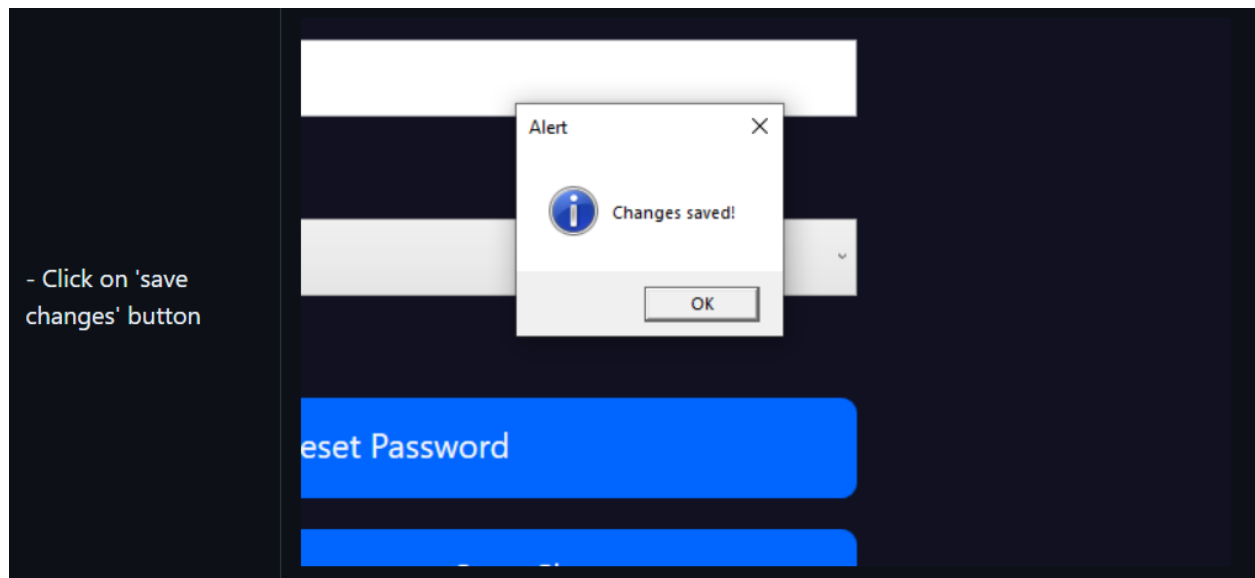


Figure 28: User Guide #3

## Implementation

Delete an account

Instruction

Example

- Navigate to 'all accounts'

All Accounts

Create Account

Account ID	Name	Account Type	Tickets
5	John Adams	Tech	0
6	Mick Smith	Tech	0
7	The Admin	Admin	0
8	Example Account	User	4
9	t t	Error - Unknc	0

- Select an account

Edit Account

Account ID

9

Name

t t

E-mail address

t

Account Type

1 - User

Password

Reset Password

Discard Changes

Save Changes

DELETE ACCOUNT

- Click on 'DELETE ACCOUNT' button

Warning!

Are you sure you want to DELETE this account?

Yes

No

Alert

Account DELETED!

OK

Figure 29: User Guide #4

## Common Feature to all Users

### Change password of current account

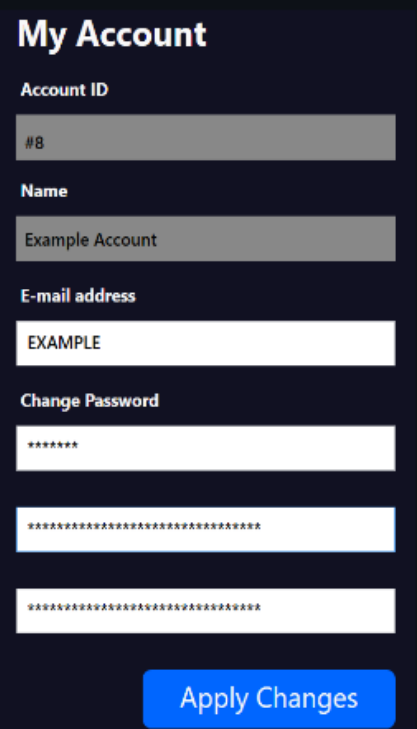
Instruction	Example
<ul style="list-style-type: none"><li>- Navigate to 'My Account'</li><li>- Enter old password</li><li>- Enter new password and confirmation of new password</li><li>- Press 'Apply Changes'</li></ul>	 <p><b>My Account</b></p> <p>Account ID</p> <p>#8</p> <p>Name</p> <p>Example Account</p> <p>E-mail address</p> <p>EXAMPLE</p> <p>Change Password</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>Apply Changes</p>

Figure 30: User Guide #5

## Create a ticket

Instructions

- Navigate to 'Create Ticket'
- Enter all fields with relevant data (Max 50 characters for title)
- Click on the submit button

Create Ticket

Create Ticket

Title

Example Ticket

Urgency

Medium

Created For

8

Created By

8

Description of problem

Here is an example description

Submit

Ticket View

Ticket #	Title	Status	Created	Updated
INCS	Example Ticket	Open	0m ago	0m ago
Caller	Created by	Urgency		
8	8	Medium		Resolve

Add Comment Here

Submit Comment

EA

Example Account

0m ago

START

Figure 31: User Guide #6




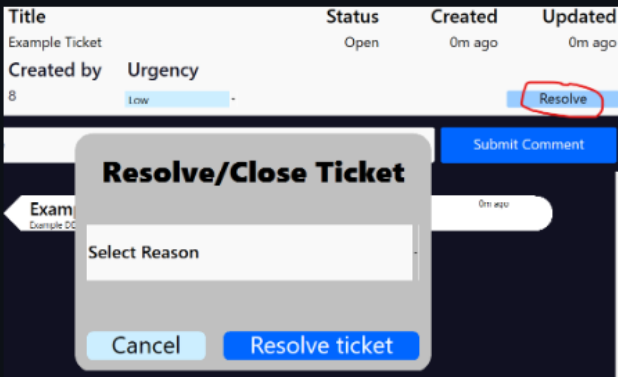

Add comment to ticket	
Instruction	Example
<ul style="list-style-type: none"> <li>- Open or create a ticket</li> <li>- Type in the comment box below the ticket info and above the current comments</li> <li>- Click on the 'submit comment' button to add a comment</li> </ul>	
Resolve a ticket	
Instruction	Example
<ul style="list-style-type: none"> <li>- Open or create a ticket</li> <li>- Click on the 'resolve' button</li> <li>- A window will pop up to select a reason</li> <li>- Select a reason to resolve/close the ticket</li> <li>- Click the 'resolve ticket' button on the pop up window</li> </ul>	
Creating a ticket on behalve of another user (i.e. a caller with a problem)	
Instruction	Example
<ul style="list-style-type: none"> <li>- When creating a ticket there is a greyed out 'created by' field and a 'created for' field which you can change.</li> <li>- By default they are the same, however you can change who the ticket is created for in cases where tickets are created on behalf of somebody else</li> <li>- To create the ticket on behalf of another user, simply change the created for input as THEIR account ID instead of yours.</li> <li>- When you do this you will still have access as you created the ticket (This field cannot be changed)</li> </ul>	

Figure 32: User Guide #7