Bachelor Thesis

Master’s in computer science 300hp

Complaint system for Health Care Center Hjärtat

Halmstad 2022-09-26
Collins Abika

HÖGSKOLAN I HALMSTAD

Preface

I would like to thank my supervisor Sundas Munir for her high-quality guidance and advice throughout this project. I would also like to thank the IT support of the Health Care Center Hjärtat for providing me with a project where I can put my knowledge into practice. And finally, I want to thank Högskolan Halmstad for providing me with the preknowledge, and being able to take on this project.

HÖGSKOLAN
I HALMSTAD

Abstract

Today the IT support could be overwhelmed with loads of requests from the clients within the Health care center Hjärtat (Also known as HCCH) through email, which is an inefficient way to store and manage information of each complaint. Tracking down the list of requests is also an issue. For the goals to ensure the achievements of expanding to new clients, IT support needs a better alternative way to manage, store, and retrieve information. This thesis reports a prototype of a complaint management system implemented for the Healthcare center Helsingborg Hjärtat. The software architect pattern in this project is Model view control and applied on the Visual studio .Net core with the programming language of C-sharp and HTML. Test results showcase the prototype of the complaint management system as an administrator, and the result of the system handles the complaint cases well. The IT support experience keeps track of the complaints more efficiently sent from the client users.

Contents

1 Introduction

Storing and retrieving information has made a lot of impact in today's society. Many different institutions today are storing and retrieving information in data storage [1]. Health care centers have made a good example of storing and retrieving data. Workers in healthcare can, for instance store and retrieve information about the patient's different attributes [2]. Vårdcentralen-Hjärtat is a health care center in Helsingborg or HCCH, Sweden. The workers in the Vårdcentralen Hjärtat commonly use computer and software programs. The majority of the workers in the health care center do not have enough technical skills to solve certain hardware or software error. When an issue occurs, the health care center sends a request for solving the issue. The IT support gets emails as their only notification for any client (also known as the workers) request. The IT support in Vårdcentralen Hjärtat is planning to expand their IT support to other major new clients, for example, schools, new healthcare centers, or even support in private sectors. To expand to new major clients, IT support needs to solve the obstacle of handling loads of complaints beforehand due to potentially being overwhelmed with loads of requests from the clients within Vårdcentralen Hjärtat through email which is an inefficient way to store and manage requests information. This thesis proposes implementing a software prototype for IT support in Vårdcentral Hjärtat.

1.1 Problem statement

When a computer issue occurs in the HCCH, a worker sends an email to IT support to complain, for example, a software program not functioning well or a network not working. When the IT support of the HCCH receives the email, the IT support provides the requested service. The IT support is planning to expand to new major clients. Getting requests and storing the complaint information from the workers or the client in email is not optimal due to the lack of monitoring and tracking. A better complaint management system is required for IT support when they expand their IT support service to the new major client by receiving, responding to, monitoring, and retrieving client complaints.

1.2 Purpose

- The purpose of the project is to improve managing, storing, and retrieving complaint information for IT support.
- Another purpose for the project is to analyze how to withstand and handle loads of complaints and request data as the IT support in HCCH Helsingborg is planning to expand to new major clients.

1.3 The goal

The goal of this project is to replace the current way of sending complaints through email with a prototype of a complaint management system. For this, a prototype of a complaint management system is implemented that is able to receive and send complaints. The prototype is a web application. The goal is also to prioritize different complaints. Another goal is to showcase a scenario of scaling the system for the IT support future extension. In order to achieve these goals, there are certain requirements that need to be fulfilled.

1.4 Requirements specifications

The project consists of different requirements which were given by the project providing the IT support of HCCH.

- Provider, the IT support in the HCCH is sole the *Admin* of the system.
- The *Admin* of the prototype web application must be able to retrieve the information from the users and update the case status when it validates and completes the complaint task.
- The frontend interface must be minimalist and non-complicated design due to the clients being non-technical.
- Prioritize complaints based on the category. Scale of preference will be available depending the acute status or the category
- Software performance must be attainable.

1.5 The outline of the thesis

- Section 1: Introduces the project, its purpose, and its goal.
- Section 2: Presents the Background of the project and gives a better understanding of the project.
- Section 3: This section presents the method adopted to implement the prototype of this project.
- Section 4: This section presents an implementation of the complaint management system prototype.
- Section 5: This section presents the details of the test cases.
- Section 6: This section presents a discussion based on the outcomes and limitations of the project.
- Section 7: This section presents the conclusion of this project.

1.6 Project limitation

The aim is not to fully deploy the prototype to the new major clients of the IT support. The complaint management system prototype is to test only a process of a complaint management system. An advanced frontend user interface is not the priority of the project. The priority in the frontend is a functional design where the client can send a complaint according to the project requirement. The web application in this project is only a test phase for the IT support before fully expanding to new major clients.

2 Background

This section presents the background details on the complaint management system, client-server model, the data that is stored, and software architecture. This section also presents similar works and literature reviews.

2.1 Complaint management system

A complaint management system is a management technique for evaluating, analyzing, and responding to complaints. Complaint management software resolves and responds to complaints and inquiries and facilitates any other feedback [3]. The Complaint Management system consists of multiple processes for dealing with grievances in the organization to solve the problem. Figure 1 describes the procedure for handling complaints. Studies of email re-finding by Elsweiler D, Baillie M, and Ruthven I [4] show that using emails as information storage and retrieval within an organization is not efficient. Other studies [5] [6] also suggest that an alternate way of storing and managing information such as client requests would be more efficient.

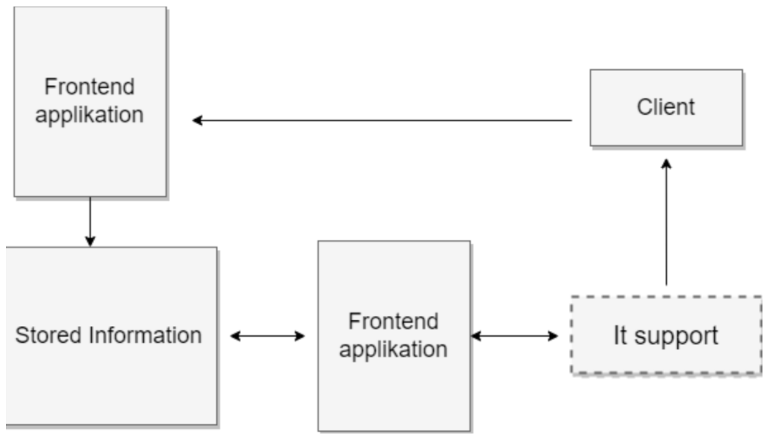


Figure 1: Complaint Management System

2.2 Client server model

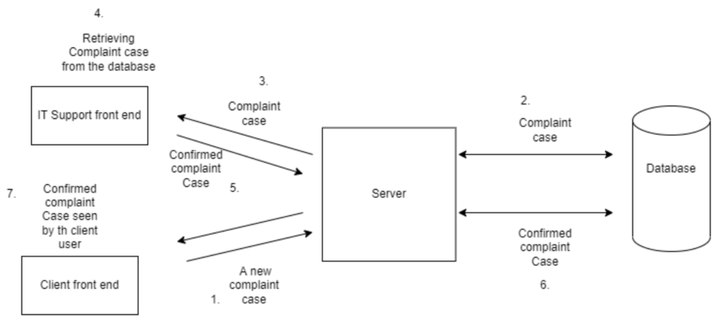


Figure 2: Client server Model

A complaint management system consists of a client-server model, a software architecture of a client, and a server (see Figure 2). The client side of the model sends requests to interact with a web application with a user interface or in other words frontend. A complaint management system consisting of the client-server model, needs a database, frontend, and back-end.

To start with the frontend, there are different categorized programming languages. The frontend can be implemented using different languages such as HTML (HyperText Markup language), CSS, and javascript. HTML is a text-based approach to describing the structured content of the complaint management system interface. It showcases how text, images, and other elements are displayed. CSS or Cascading Styles Sheets is the language that describes how the elements can be displayed on the screen. Javascript is the programming language that enables interaction with a user. With the three different programming languages of a web application, the first element of the complaint management system frontend enables user clients to interact.

Without the second element of the complaint management system, which is the back-end. The back-end consists of two-element of the complaint management system, the server, and the third element is a database. The server responds to the request by performing the operations according to what is requested. The client-server uses protocols such as HTTP (Hypertext Transfer Protocol) to transfer text, images, and videos. In a complaint management system, the client is then able to send the request to the server, and the operation will be performed. A popular language that is used for a server is PHP(Hypertext Preprocessor) [7] [8].

2.3 Database

The database is the third element of the complaint management system that stores all the data and information. In a complaint management system, when a client sends a complaint, it is important to store information with different attributes from the client on a large scale. The importance of handling large data is due to the HCCH IT support plan for their expansion. A better structure of incoming complaints can be exhibited. A traditional database model is a relational database.

2.3.1 Relational databases

In relational databases, the data in these databases are organized into predefined categories using a series of tables. A table is made up of rows and columns, with each column containing data for a given category and each row containing an attribute of that data determined by the category. To scale up with a relational database approach requires using a larger computer with loads of memory space. This approach is called vertical scaling. Structured Query Language SQL is a programming language that provides access control, manipulation, and defining data information, [9]. Operations in SQL called CRUD.

2.3.2 CRUD

CRUD(Create, Read, Update, Delete) is a key feature that operates in relational databases. It plays a significant role in a complaint management system. The operation" creates", creates new data that adds and stores it in the database when a user client sends a complaint. Read operation is when the administrator retrieves the information of the client with the complaint message. The operation "update" is the data that updates by the client and executes the task from the database. "Delete" is when the operator removes the data from the database [10]. The operations manipulate and fetch the data to the server as an object. The code structure in the server uses a software architecture called Model View Control.

2.4 Model View Control

The model is divided into three layers which contain Model, View, and Controller. The model processes all function logic written in code and fetches data from the database. The *View* updates the view or the information displayed to the user. The *View* contains mainly buttons and information that is suitable for user interface logic. The *Controller* interfaces between the *View* and the model layer. When the user of the prototype interacts with the system, the *Controller* requests the functions or data from the model (see figure 3).

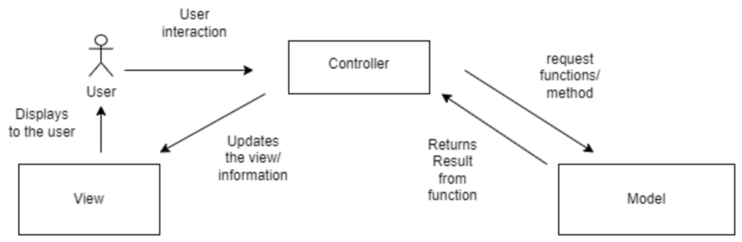

User interaction
User
Controller
request functions/ method
Displays to the user
Updates the view/ information
Returns Result from function
View
Model

Figure 3: Model View Control

2.5 Scalable complaint management system

A company that is planning to expand its service to more clients must consider scaling. Scaling is defined as a performance not being affected when it is moving from a smaller operating system to a larger operating system. It also measures how much a computer system can handle a high load of users. An organization that can scale will be prepared for high user load and even expansion. For the IT support in the HCCH's case, expanding to new major clients and being able to scale and knowing when to scale up is optimal [11]. A new major client would indicate more traffic for a web application.

2.6 Literature review

This subsection presents statements from different literature that motivates the relevance of the problem that is to be solved in this project.

According to previous reports, increasing the number of incoming emails of complaint requests is inefficient if an organization needs to scale. [12] [13]. It leads to potentially missing track of the problem request and what category the issue is about. A case study from Mercu Buana University suggests software management that uses ticketing is a better alternative [13]. A ticketing feature is a list of client requests that are organized to keep track of the information and consist of status [14]. A helpdesk software management with a ticketing feature was implemented to facilitate dividing and sorting incoming complaint requests into different categories as opposed to receiving them in a single email or from phone calls. The report [13] concluded that the implementation of helpdesk software is efficient in keeping track of incoming complaints from clients.

According to another report [12], using email to receive client requests is impractical. A better alternative way is that requests coming from the clients are recorded and stored. A feature that automatically records and stores the data from the clients is called ticketing. In the report, a form was used to analyze what features can be used in the help desk complaint management system. The results showcase that prioritizing different incoming data and updating status in a help desk complaint management system is significant [12].

2.7 Related work

This subsection presents similar work and key takeaways that are useful for the complaint management system in HCCH. The work does not specifically showcase a complaint management system in a healthcare center but showcases examples of handling and managing client complaint requests.

A complaint management system similar to this project is implemented in a Tertiary institution project as a solution for the students to file concerns and preserve reliable records of such complaints [15]. The Waterfall Methodology was used as the research methodology for this project. The Database Management System (DBMS) is MySQL, while the Integrated Development Environment is Visual Studio Code (IDE). The test results demonstrated that complaints stored in a database make it easier to retrieve respective rather than the student writing a formal complaints letter to the Head of Department. A Key takeaway from this is the waterfall methodology. The waterfall methodology focuses on each key point to be achieved before aiming to achieve the next one, such as planning, designing, developing, and testing [15].

Another study presents a software solution with a similar concept of a complaint management system for workplace PC troubleshooting [16]. The system includes some functions, such as issue status, priority, and storage. The system was built and developed following the organization's standards. The result of the project demonstrates that the system satisfies the needs of the information technology department by including capabilities such as recording solutions for future reference, text chat, and sending text messages, among others. One of the key takeaways from this project is how the Entity-Relationship (Also known as ER) diagram was designed and how tables are relating to one another. The helpdesk management system contains employees, technicians, and a user account table. Many employees can send a complaint to the technician which results in a "many too many" relationship, An employee can only have one user account which turns out to be one to one relationship. Knowing the type of relationship between the tables helps in understanding and designing an ER diagram for a database of the complaint management system in HCCH. Another takeaway is using priority as a feature for a complaint management system. Such a feature has the advantage to prioritize the most urgent issue firsthand before handling other issues with less urgency. It results in saving time and efforts [16].

2.7.1 Key takeaways from the related works

The common nominator of a solution with the related works in sections 2.8.1-2.8.3 for complaint management is a database and a user interface. A takeaway is categorizing the incoming records for them to be more effective in tracking and handle loads of information.

3 Methodology

The methodology in this project carries out with the waterfall methodology [17]. The method in this project of implementing a complaint system splits into different phases. The waterfall model approach is a methodology known for software development and engineering.

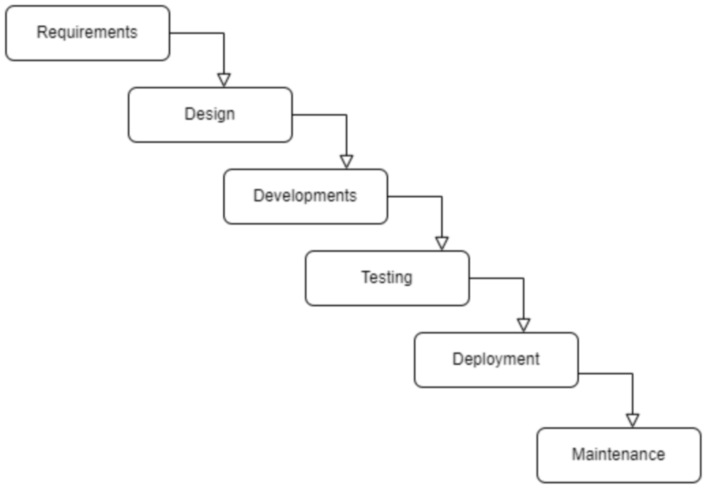


Figure 4: Waterfall model

3.1 Requirements

There are four requirements in this project.

- The first requirement is to ensure that only the IT support has access to the admin portal.
- The second requirement is to ensure that the complaint management system performs CRUD operation where the *Admin* is able to retrieve and update the complaint status that is sent from the client.
- The third requirement is that the complaint list on the *Admin* side sorts in order of priority. The priority depends on the category of the complaint sent from the client.

- The fourth requirement is a good design of the user interface. This requirement is for the client whose knowledge to IT is low.

These main requirements can ensure the achievement of the goals of this project.

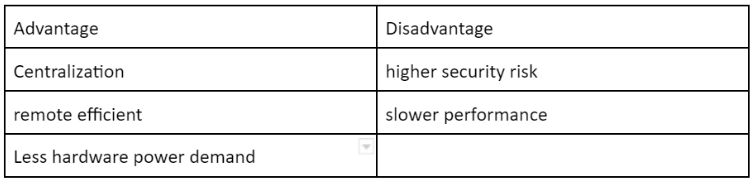
3.2 Design

3.2.1 Choice of method

In this project, a full-stack web application is a choice for implementing a complaint management system. There are several ways to implement a complaint management system. One way to implement a complaint management system is a stand-alone desktop application. The desktop app is more secure if it runs offline which makes it more difficult for hackers to access the software program. More powerful hardware components are required to run a desktop application which makes the speed performance faster. A desktop is capable of running offline, which makes privacy safe using the app. The disadvantage of the desktop application is downloading process and manual upgrade for every computer. Desktop apps consume more memory space than web applications. Webb application on the other hand is more centralized due to one server that is running the web application program on computers from all over the world can access and use through the website. An advantage of the centralization system is that all the computers do not need to go through downloading a program. A web application is more remote efficient than a desktop application which is what the HCCH is aiming for. Implementing a complaint management system on a web application also comes with some downsides. The downsides of web applications are that higher security risk due to web applications only running online, which makes it more likely for hackers to access the web app than a desktop application.

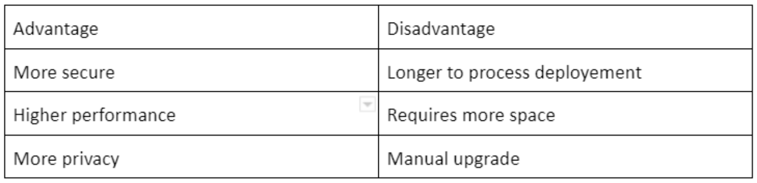
Another way to implement a complaint management system is a mobile app. Mobile app usability is higher than web application, according to research [18]. Mobile applications run both offline and online compared to web applications with faster execution time. The downside of implementing a mobile app instead of a website comes with a longer process to deploy, adapting the operating system's latest update [18] [19].

In conclusion, a web application is a choice for implementing a complaint management system due to a centralization system that can update the application without the clients updating the web application remotely. It is also due to the IT support's future expansion plan where new clients can access the web app through a web browser without downloading the program, which is time-consuming [20]. An analysis of the pros and cons of each choice is included in figures 5-7.



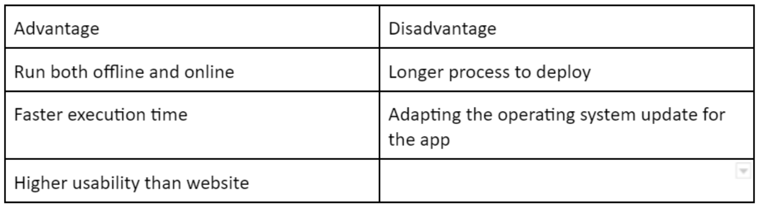
Advantage	Disadvantage
Centralization	higher security risk
remote efficient	slower performance
Less hardware power demand	

Figure 5: Advantage and disadvantages web application [20]



Advantage	Disadvantage
More secure	Longer to process deployement
Higher performance	Requires more space
More privacy	Manual upgrade

Figure 6: Advantage and disadvantage of desktop applications compared to web application [20] [21]



Advantage	Disadvantage
Run both offline and online	Longer process to deploy
Faster execution time	Adapting the operating system update for the app
Higher usability than website	

Figure 7: Mobile application advantage and disadvantage compared to web application [18] [19]

The prototype works in the form of a full-stack web application. It consists of three different programs that work as a combination for the application to function [22]. A Full-stack web application is based on the client-server model (section 2.2).

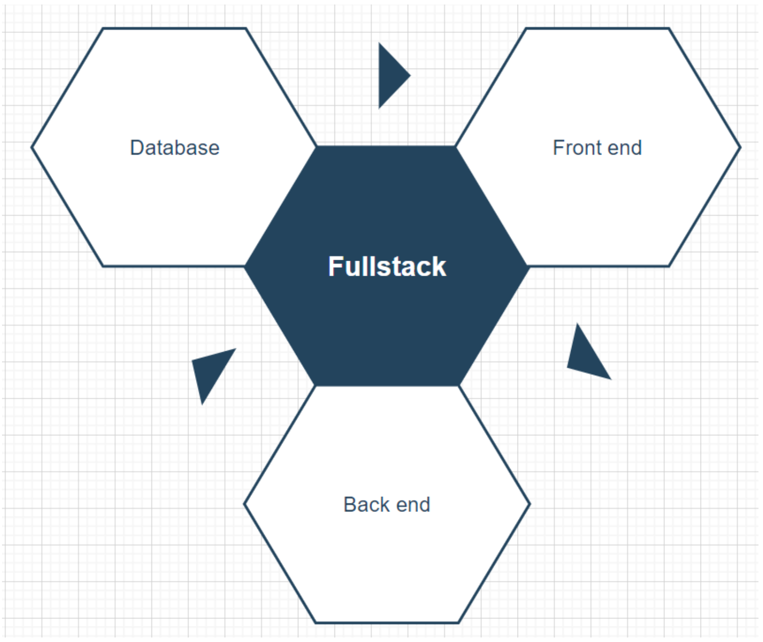
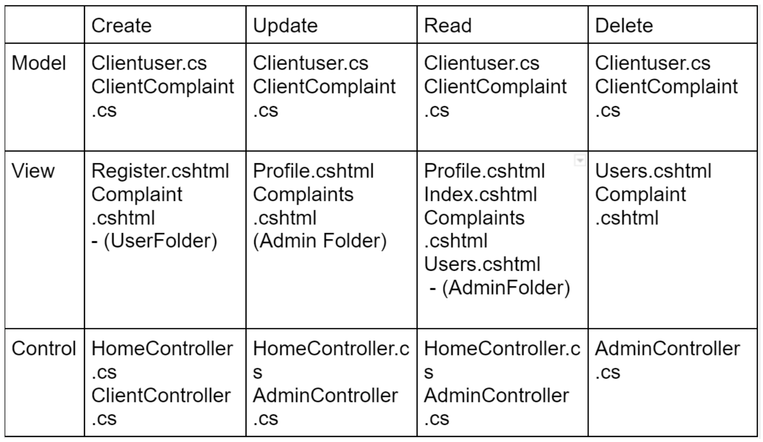


Figure 8: The three key elements of the prototype

3.2.2 Model View Controller

In the visual studio, a .Net core project was created with a model, view, and controller, each of them as a folder. In the model folder, three model classes were implemented. The classes in the model folder are clientcomplaint.*cs*, *Clientuser.cs* and *UserAdmin.cs* file. The object-orient classes receive the fetched information data from the database. The object classes contain the same attributes in the tables being fetched from the database. Each of the object-oriented classes is meant to be fetched from the three different tables in the database. In the Controller, four controller classes were implemented. *AdminController*, *ClientController*, *ComplaintController* and Homecontroller. Homecontroller is the controller class for the client user. The classes consist of methods that redirect. View files are consist of three different layouts (see figure 9).



	Create	Update	Read	Delete
Model	Clientuser.cs ClientComplaint .cs	Clientuser.cs ClientComplaint .cs	Clientuser.cs ClientComplaint .cs	Clientuser.cs ClientComplaint .cs
View	Register.cshtml Complaint .cshtml - (UserFolder)	Profile.cshtml Complaints .cshtml (Admin Folder)	Profile.cshtml Index.cshtml Complaints .cshtml Users.cshtml - (AdminFolder)	Users.cshtml Complaint .cshtml
Control	HomeController .cs ClientController .cs	HomeController.c s AdminController .cs	HomeController.c s AdminController .cs	AdminController .cs

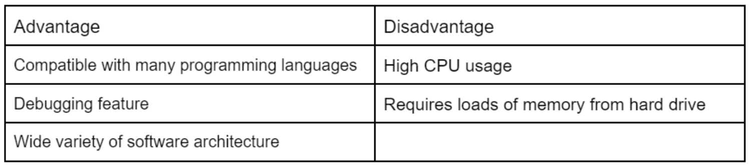
Figure 9: Classes in the MVC in the crud operation

3.3 Development environments

A compatible and suitable development environment is important when developing the prototype [23]. This subsection showcase's brief analysis of the difference between the development environments and database management systems for this project.

3.3.1 Visual Studio

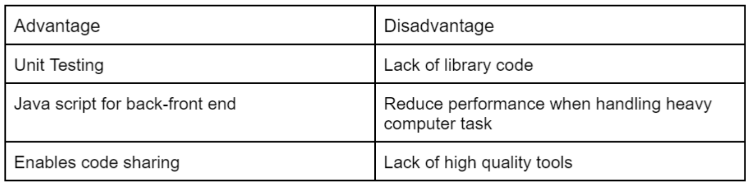
Visual Studios is a full-stack development environment with the possibilities of programming web and databases. The development environment is a visual studio implemented by Microsoft that provides multiple options of programming language for any developers to use. A development environment that supports multiple programming languages makes it flexible. A feature such as debugging is an advantage due to making it faster to find the error in the process of troubleshooting. Another advantage of visual studio is the wide variety of supported features that combine web and native programming languages. The downside of visual studio is high CPU usage and loads of memory requirements from a hard drive [24].



Advantage	Disadvantage
Compatible with many programming languages	High CPU usage
Debugging feature	Requires loads of memory from hard drive
Wide variety of software architecture	

Figure 10: Advantages and disadvantages of Visual Studio [24]

3.3.2 NODE.JS



Advantage	Disadvantage
Unit Testing	Lack of library code
Java script for back-front end	Reduce performance when handling heavy computer task
Enables code sharing	Lack of high quality tools

Figure 11: Advantages and disadvantages of Node JS [24]

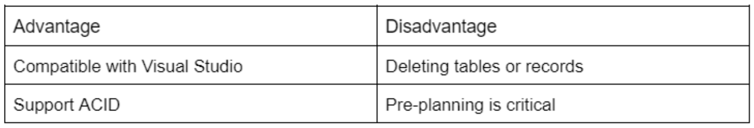
NODE.JS is a development environment built on Chrome notoriously for a mobile application that also suites for web applications [25]. The only optional programming language when the developers are using NODE.JS is Java script. A run time environment provided by NODE.JS is implemented in C++ called V8. V8 run time environment consists of good code management and compilation to its native code. The Node.JS library includes an HTTP server module that can be extended with packages. The downside of Node.JS is that it is compatible with none relational databases but is not compatible with relational databases. [26].

3.4 Comparison between database development environments

This section presents comparisons between database development environments and optimal choice for the prototype.

3.4.1 Microsoft structured query language

MSSQL (Microsoft structured query language) is a relational database platform developed by Microsoft. Its primary function as a database server is storing and retrieving requests from other web or software products. An advantage of using MSSQL as the database platform being compatible with visual studios. It leads to more flexibility between the MSSQL and the visual studio during the development of a web application. ACID is a property that ensures valid transactions for the database which MSSQL acquires. Another upside is that the server can run on localhost and a network. The downside is that removing a table is not flexible. Removing a table means removing all relations between other tables. This downside of the MSSQL indicates good planning is required before implementation is critical for time-consuming if an error were to occur.

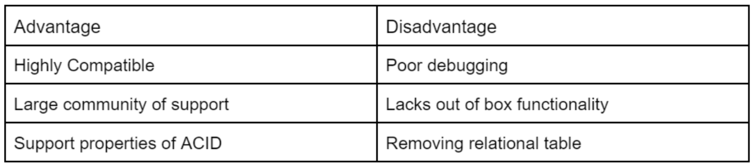


Advantage	Disadvantage
Compatible with Visual Studio	Deleting tables or records
Support ACID	Pre-planning is critical

Figure 12: Advantage and disadvantage of using Microsoft structured query language

3.4.2 MySQL

MySQL is an Oracle-developed relational database management system based on structured query language (SQL). MySQL's primary objective is to be compatible with a wide range of technologies and architecture. It makes MySQL open to different solutions. Being compatible makes it work with a wide range of operating systems and languages, including PHP, PERL, C, C++, JAVA, and others. Since it is open-source, there is a large community of developers that can help. MySQL maintaining the integrity of data through atomicity, consistency, isolation, and durability (also known as ACID) is also an advantage. A database platform with ACID will ensure preserving the consistent state of data. The downside of the relational database management system is its lack of debugging. This downside of MySQL could potentially lead to more time consumption during troubleshooting if an error were to occur. MySQL lacks out-of-box or so-called built-in functionality, which means it does not come with native features after installation. It makes the MySQL plugin dependent. A similar downside to MSSQL in section 3.3.5 is good planning requirement before implementation is critical for time-consuming if an error were to occur due to it being a relational database platform [27] [28].



Advantage	Disadvantage
Highly Compatible	Poor debugging
Large community of support	Lacks out of box functionality
Support properties of ACID	Removing relational table

Figure 13: Advantage and disadvantage of MySQL

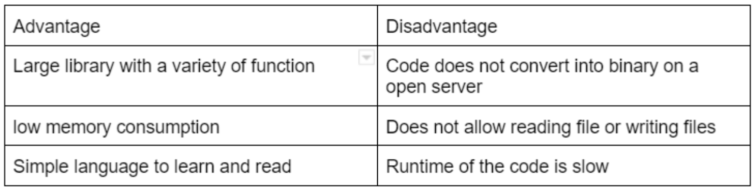
3.5 Programming language

In this subsection, the two programming languages JavaScript and C-sharp, are compared for the optimal choice for the prototype.

The advantage of JavaScript is that the language works on different platforms, such as mobile applications and also console applications. This language does not take much memory and has good speed and potential when less memory is used. JavaScript utilizes a large amount of library which simplifies a programmer's [29].

The disadvantage is that when the code is stored on an open server, the code is not converted to binary form as C-sharp does. This leads to reduced security and a greater chance that hacking can take place. When it comes to running code, JavaScript is very tough, and a long delay can occur, especially if it is a large piece of code. This is because the language does not utilize multiple processors. JavaScript also does not allow the reading of files and does not write files either for security reasons.

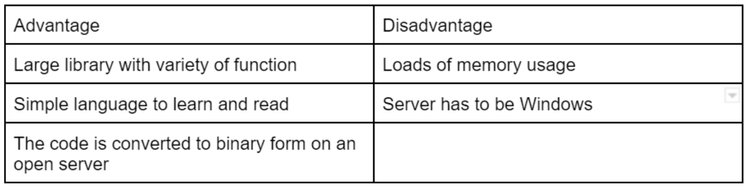
C-sharp is the language most used in Visual Studio. The language is easy to learn and easy to read the coding. C-sharp has a large number of features, more than JavaScript. A big advantage is that C-sharp integrates very well with



Advantage	Disadvantage
Large library with a variety of function	Code does not convert into binary on a open server
low memory consumption	Does not allow reading file or writing files
Simple language to learn and read	Runtime of the code is slow

Figure 14: Advantage and disadvantage of JavaScript

Windows. It was created specifically for Windows. The language is also efficient, and the process from development to production can take place very quickly. The language is similar to JavaScript, and it is very easy for a programmer to learn the language. Another advantage of C-sharp is that well when the code is stored on an open server, it ends up there in binary form. The disadvantages are that C-sharp is part of the .NET framework, so the server running must be Windows. The language is not as good as JavaScript in developing applications for multiple platforms. C-sharp language develops apps for the Windows platform, and a lot of memory is also used in C-sharp.



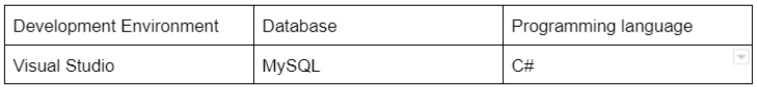
Advantage	Disadvantage
Large library with variety of function	Loads of memory usage
Simple language to learn and read	Server has to be Windows
The code is converted to binary form on an open server	

Figure 15: Table of advantage and disadvantage with C-sharp

3.6 Choices of development tools

With a sample of advantages and disadvantages of programming language and development environment, the choice of development environment for the back-end server is Visual studio. Visual Studio enables the usage of .Net core. .Net core is a modern, open-source, multi-platform, and multi-purpose development framework for creating modern, quick, and scalable applications .NET Core is open source and supported by a large community [24]. The choice of programming language is C-sharp due to the simplicity and knowledge before the project. C-sharp supports a large number of library functions. The choice of the database management system is MySQL due to its being highly compatible, and it makes

MySQL open to different solutions. Being compatible makes it work with a wide range of operating systems and languages such as C-sharp.



Development Environment	Database	Programming language
Visual Studio	MySQL	C#

Figure 16: Choices of development tools

3.7 Testing strategy

The results of the testing strategy are based on the goals that are set in the introduction section, which measures if the achievements are reached. The testing strategy consists of four different tests.

- The first test is login in as admin. A user registers in the system under testing by entering the user name and password in the admin portal. If the user is not able to log in as the *Admin*, then the first requirement specification is achieved.
- The second test is IT support as an *Admin* retrieving complaints from client users of the complaint management system. The first step in the second test is creating a user account and sending a complaint. To confirm the first step is working, the database updates in the client table and the complaint table. The second step in the second test is logging in as *Admin* and checking if the same complaint in the database is also on the admin page. The third and final step in the second test is updating the complaint status and checking in the database and the user account if the complaint status is updating. With the three steps of the second test working, the second requirement is achieved.
- The third test consists of testing the priorities of the complaints. The first step of the third test is to send three or four complaints from the user and check if the complaints sort depending based on the category. The complaint with the highest priority is at the top of the complaint list. If the complaints with the highest priority based on the category are at the top of the list, then the third requirement specification is achieved.

- The fourth test is to examine the user interface. The IT support workers are two reviewers checking each page of the complaint management system. The review is rated on a scale between 0-10. The rating formula is shown below:

  $$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n)$$

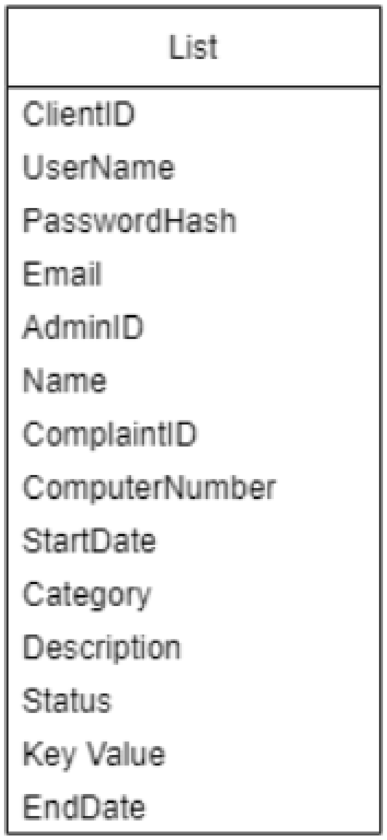
  "n" represents the number of pages, "x" is the rating score for each page, and the variable $\bar{x}$ is the result. If the results correspond to 100 percent rating success, the fourth requirement specification is achieved.

Achieving four requirement specifications in the tests ensures the goals of this project.

4 Implementation

4.1 Normalization of database

Before implementing the database, designing the table by normalizing it is necessary for a relational database. The table is designed from the information that is necessary for the IT support to receive from the employees in HCCH, and the table is also designed for the employees to receive information on the complaint case status. The table contains a total of 16 different attributes, as shown in figure 17.



List
ClientID
UserName
PasswordHash
Email
AdminID
Name
ComplaintID
ComputerNumber
StartDate
Category
Description
Status
Key Value
EndDate

Figure 17: The complaint case table

Due to both employee clients and IT support in need to receive information for the complaint case, the first normalization form (also known as 1NF) is not enough. 1NF is a table with non-repetitive columns, and 2NF is 1NF with separate tables with different attributes depending on the primary key. Therefore the 1NF table was developed to 2NF, which contains three tables. The three tables are client user, complaint, and *Admin* table (see Figure 18). A Client user will be able to send multiple complaints. Therefore the table from client user to complaint table is on to many relations. An *Admin* can receive and read a list of complaints from different senders, which makes it on to many relations from the table *Admin* to the complaint table. The IT support *Admin* will read the description of the complaint case and update the status when the task is received and completed. The table relates through the client ID and Admin ID due to the attributes being the primary key in each of the tables.

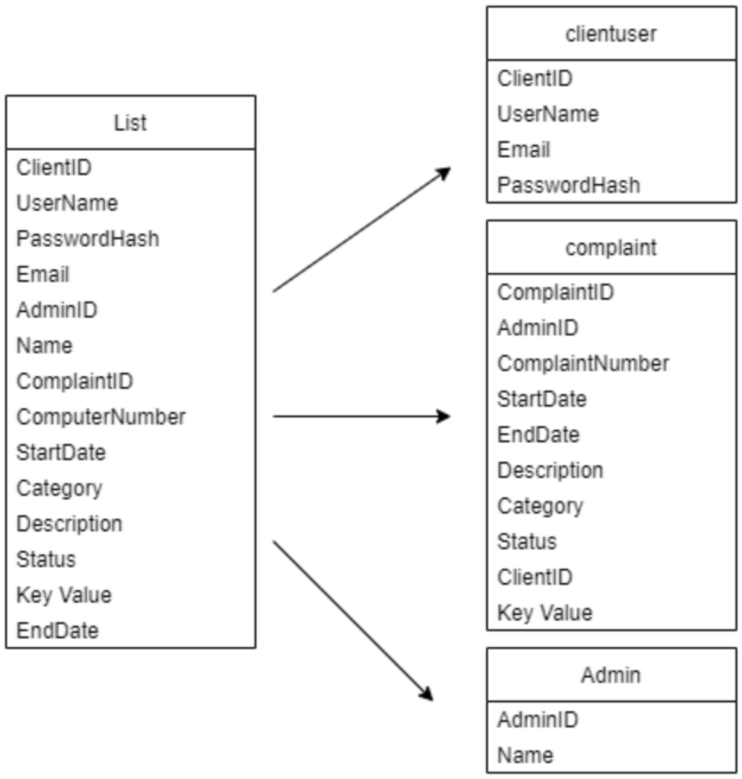


Figure 18: Complaint case table in 1NF to 2NF

4.2 Frontend design

The frontend contains a total of 12 different pages with six pages each for both the *Admin* and a client user. The client user can log in with the correct username and password. If the client user does not have an account, the client can register and then be able to log in to send complaints. When the *Admin* logs in, a register option is available on the start page to prevent any user to register as an admin. On the start page, there is a login option for the *Admin*(see Figure 19).

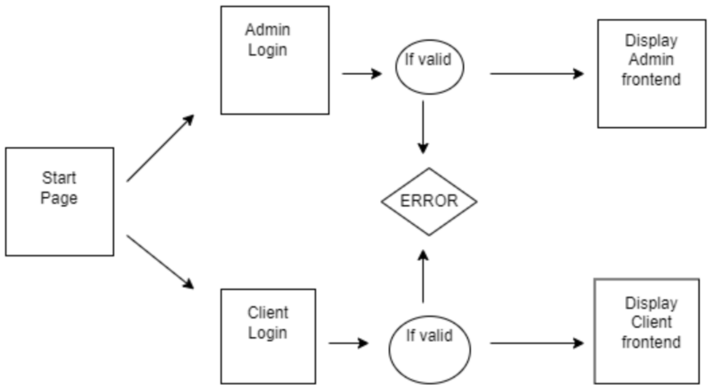


Figure 19: Admin and client login

4.2.1 Client side frontend

The client side frontend contains six pages. The first page starts with the login page for the client user. If the user of the client side does not have an account, then the registration route will be the option. As the client is logged in, the home page contains a list of complaints that were sent from the same client user. The first route is the next page which is where the client can fill in new complaints and return to the homepage to see the status of the complaint. The second route is the Update page, where the client user can update their profile information (see figure 20).

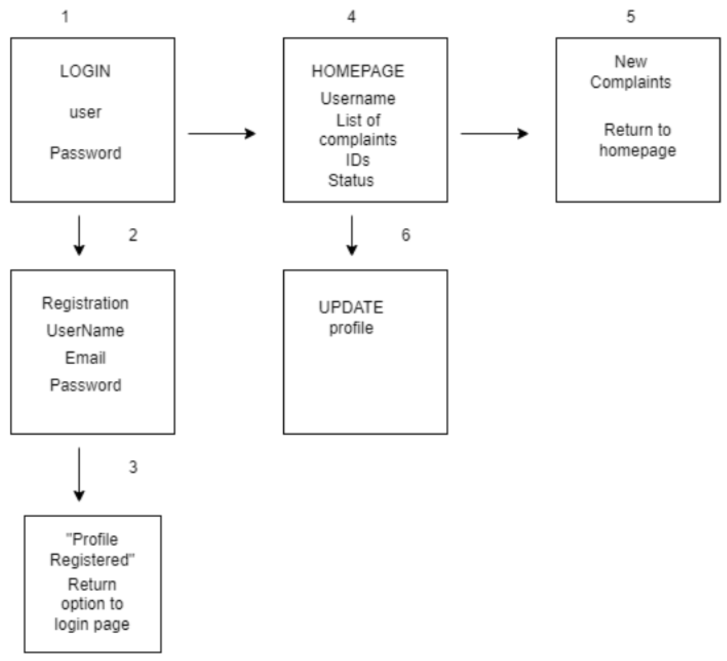


Figure 20: Client frontend route map view

4.2.2 Admin side frontend

The admin frontend contains six pages with two routes diverging from the second page. The first page of the first routes contains a list of complaint IDs. The second page of the first route contains a selected complaint ID with details. On the second page of the first route, the admin updates the status of the complaint ID when the complaints have been validated, processed, and completed. In the second route, the first page contains a list of client user IDs with a username. When a client ID is selected, the second page of the second route will contain the client user detail and a list of complaints sent by the selected client ID. As an Admin, client users are able to be deleted on the same page (see figure 21).

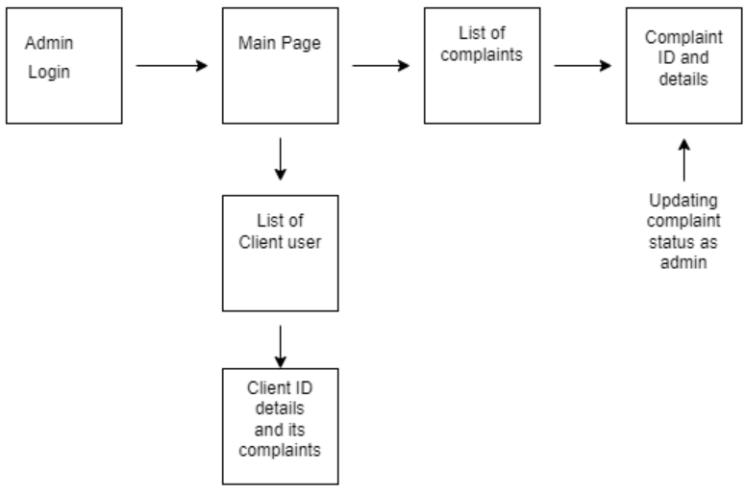


Figure 21: admin frontend route map view

4.3 Implementation of database

The database is developed in MySQL workbench. The first step is designing an ER model. The tables designed in the ER model are the client user table, complaint table, and admin table. Relation between client table and complaint falls into the category of one to many. The relation category is due to one client user in HCCH being able to send many complaints. The relation between the Admin table and complaint also falls into the category of one many relations due to one admin as able to receive multiple complaints accordingly (see figure 22).

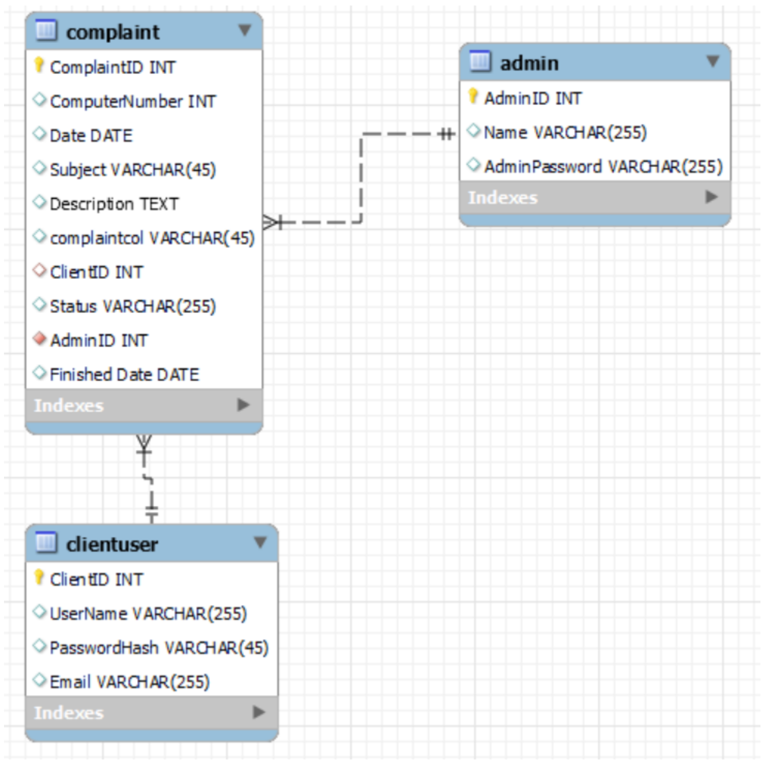
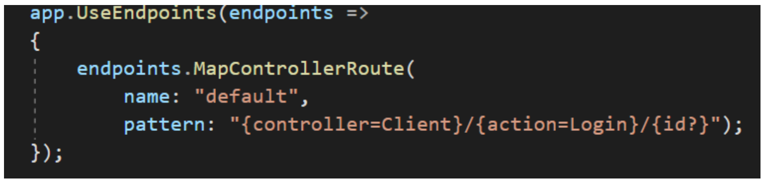


Figure 22: ER model

4.4 Start page

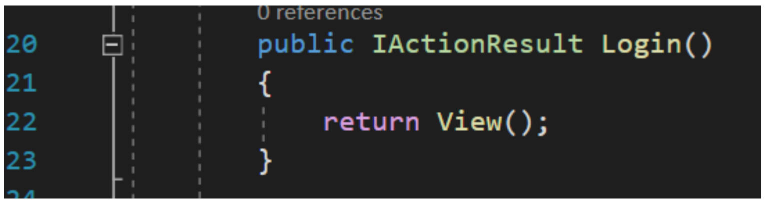
As the web application runs and displays the login.cshtml. file as the first page, the map controller route in the startup.cs file configures for the login.cshtml file to be the first file (see figure 23).



```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Client}/{action=Login}/{id?}");
});
```

Figure 23: implementing login file as start page

The *ClientController* is called and is specified to use the action login method, which then returns the view for the login page (Figure 24).



```
0 references
public IActionResult Login()
{
    return View();
}
```

Figure 24: Action Login method

Action Methods are responsible for executing requests and generate responses to them. For a user to have a user account, a navigation link to the registration page or the *register.cshtml* file is declared ”asp-controller” to ”Client” and asp-action to ”Register” in the login.cshtml file. The asp-controller uses the *client-Controller.cs* to find and execute the Iaction Register method. There is two register action method in the client controller class where one has an input for the clientuser object. In this case, the asp-action calls for the one without the client user input due to the navigation link not providing any input. Register method then returns view *Register.cshtml* file, which switches the page from the login to the next page.

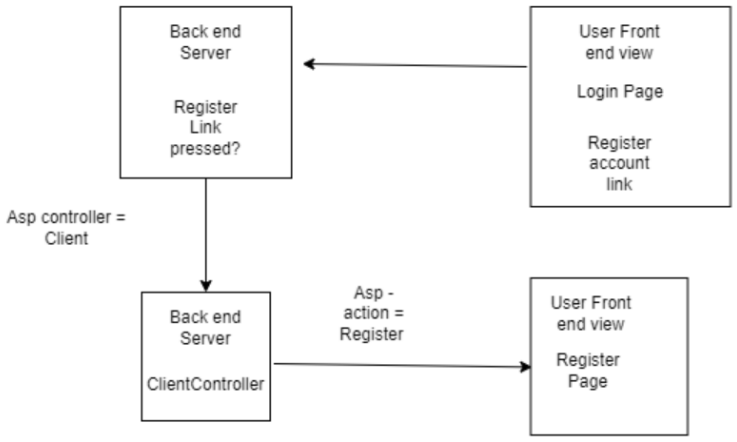
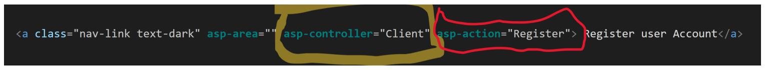
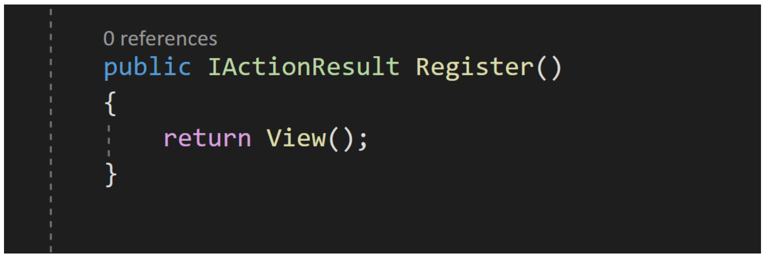


Figure 25: Navigation



```
<a class="nav-link text-dark" asp-area="" asp-controller="Client" asp-action="Register"> Register user Account</a>
```

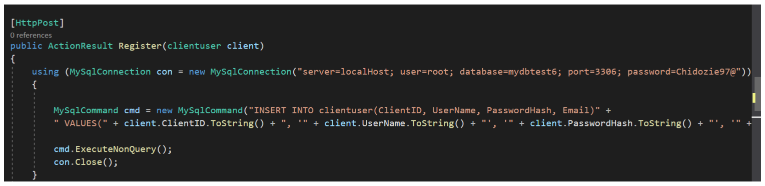
Figure 26: Navigation link in Login view file



```
0 references
public IActionResult Register()
{
    return View();
}
```

Figure 27: Register action method

The registration page ”Html.beginForm” is implemented with three different inputs for each attribute of the object the client user is to receive from the text inputs written by the user. When the ”Html.BeginForm” receives the inputs, The *Register* action method with input for client user object from the controller client is then called. In the *Register* action method, MySQL connection variable ”con” opens a connection to the database. The connection is provided with a string written in SQL language. When the insert occurs, the client user attributes are converted to Strings and added as values in the SQL string code. When the data is added to the database, and the user returns to the first login page being able to log in.



```
[HttpPost]
0 references
public ActionResult Register(clientuser client)
{
    using (MySqlConnection con = new MySqlConnection("server=localHost; user=root; database=mydbtest6; port=3306; password=Chidozie97@"))
    {

        MySqlCommand cmd = new MySqlCommand("INSERT INTO clientuser(ClientID, UserName, PasswordHash, Email)" +
        " VALUES(" + client.ClientID.ToString() + ", '" + client.UserName.ToString() + "', '" + client.PasswordHash.ToString() + "', '" +

        cmd.ExecuteNonQuery();
        con.Close();
    }
```

Figure 28: Register action method

4.5 User logged in

On the log-in page, two text input was implemented for the user to type in the user name and the user password. The inputs are then sent to the html client user object, specifically to the password and user name attributes. From there, the html begin-form navigates to the *ClientController*, which in turn to the action method *Login2*. The *Login2* method receives the client user object as input. The SQL code in the *Login2* method executes and searches for the same data in the database. If the data matches, the http session uses the client user object name attribute that matches with the database for the next page to be displayed. If the text input from the user does not match the *Login2* method will return an error warning as an invalid account. Figure 29 showcases the scenario of the user login.

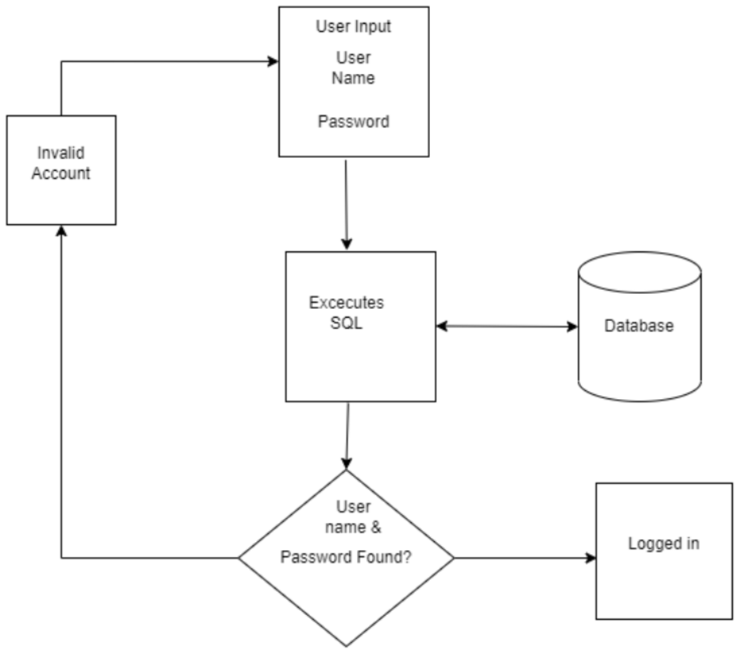
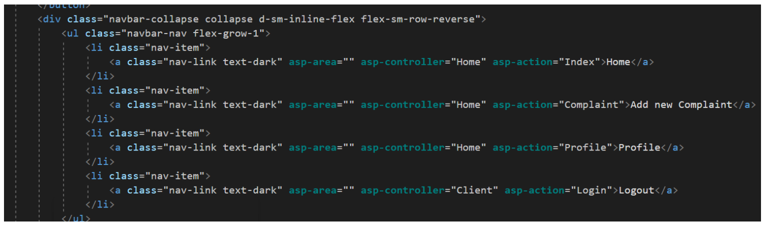


Figure 29: Flow chart of user login

4.5.1 Sending complaint

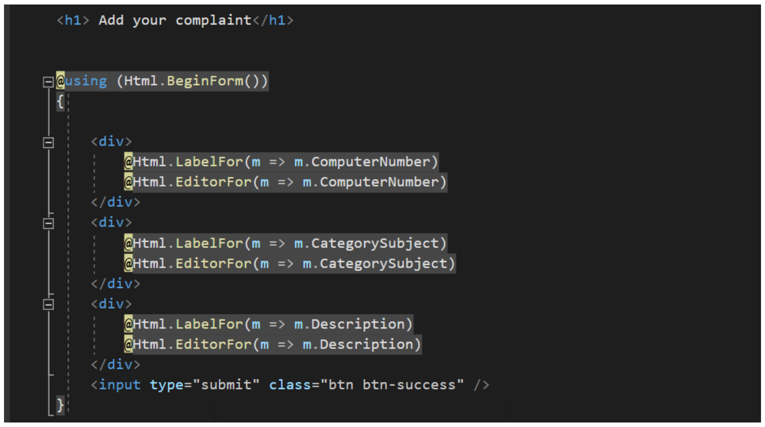
As the user is logged in for it to add a complaint, a navigation bar with a navigation link to the complaint file is implemented in the Layout file(see figure 30).



```
</button>
<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Complaint">Add new Complaint</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Profile">Profile</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Client" asp-action="Login">Logout</a>
        </li>
    </ul>
```

Figure 30: Layout of the navigation bar user logged in for the client

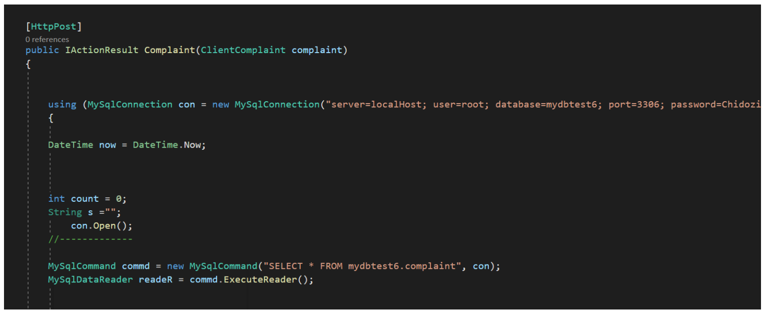
The complaint consists of implemented text inputs for complaints that match the *clientcomplaint* object attribute. The same method for creating a client user object provides the in complaint.cshtml file. The text input sent is sent to the "*clientcomplaint*" object attributes which are then navigated to the *Homecontroller* file.



```
<h1> Add your complaint</h1>


@using (Html.BeginForm())
{

    <div>
        @Html.LabelFor(m => m.ComputerNumber)
        @Html.EditorFor(m => m.ComputerNumber)
    </div>
    <div>
        @Html.LabelFor(m => m.CategorySubject)
        @Html.EditorFor(m => m.CategorySubject)
    </div>
    <div>
        @Html.LabelFor(m => m.Description)
        @Html.EditorFor(m => m.Description)
    </div>
    <input type="submit" class="btn btn-success" />
}
```

Figure 31: Text input to the object

In the "*Homecontroller*" file, the action method Complaint is called and executed by an SQL code that stores the information in the database.

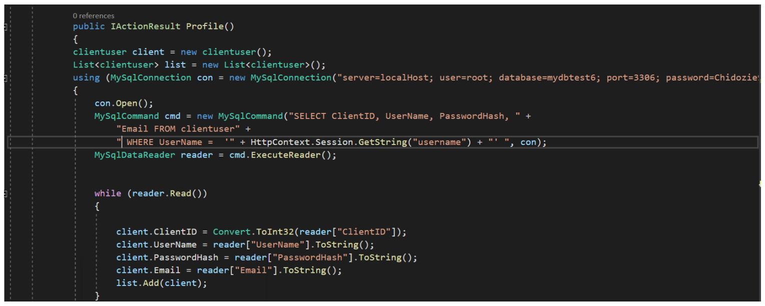


```
[HttpPost]
0 references
public IActionResult Complaint(ClientComplaint complaint)
{


    using (MySqlConnection con = new MySqlConnection("server=localHost; user=root; database=mydbtest6; port=3306; password=Chidozi
    {

    DateTime now = DateTime.Now;



    int count = 0;
    String s ="";
        con.Open();
    //-------------

    MySqlCommand commd = new MySqlCommand("SELECT * FROM mydbtest6.complaint", con);
    MySqlDataReader readeR = commd.ExecuteReader();
```

Figure 32: *Client Complaint* object executed in SQL

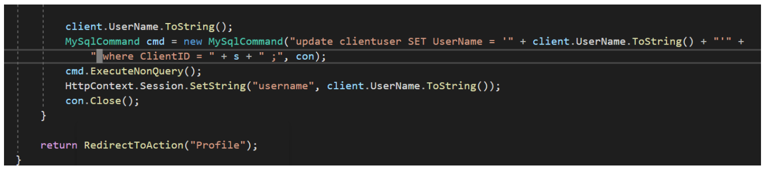
4.5.2 Update profile

In figure 33, the profile navigation link directs the user to the profile page the same way as the previous method. For a profile page to present the profile of a user that consists of account details, an SQL code was used in the action profile to select specifically the user's account detail from the database



```
0 references
public IActionResult Profile()
{
clientuser client = new clientuser();
List<clientuser> list = new List<clientuser>();
using (MySqlConnection con = new MySqlConnection("server=localHost; user=root; database=mydbtest6; port=3306; password=Chidozie
{
    con.Open();
    MySqlCommand cmd = new MySqlCommand("SELECT ClientID, UserName, PasswordHash, " +
        "Email FROM clientuser" +
        " WHERE UserName =  '" + HttpContext.Session.GetString("username") + "' ", con);
    MySqlDataReader reader = cmd.ExecuteReader();


    while (reader.Read())
    {

        client.ClientID = Convert.ToInt32(reader["ClientID"]);
        client.UserName = reader["UserName"].ToString();
        client.PasswordHash = reader["PasswordHash"].ToString();
        client.Email = reader["Email"].ToString();
        list.Add(client);
    }
```

Figure 33: Selecting user account detail from the database



```
        client.UserName.ToString();
        MySqlCommand cmd = new MySqlCommand("update clientuser SET UserName = '" + client.UserName.ToString() + "'" +
            " where ClientID = " + s + " ;", con);
        cmd.ExecuteNonQuery();
        HttpContext.Session.SetString("username", client.UserName.ToString());
        con.Close();
    }

    return RedirectToAction("Profile");
}
```

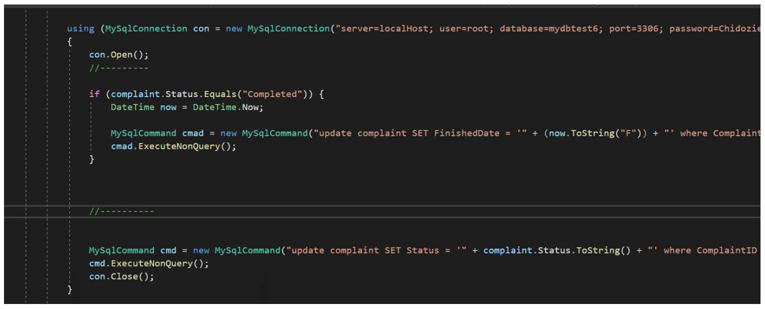
Figure 34: Implementation of Updating profile from the user

4.6 Admin

On the login page at the bottom page, a navigation link is implemented, which navigates to admin *login.cshtml* page. In the admin login, text inputs are implemented for the admin object attribute. In that case, the client user object cannot log in at the admin frontend view. When the *Admin* is logged in, the admin login page uses the admin layout view compared.

4.6.1 Complaints

The navigation link on the navigation bar is implemented for the *Admin* user to be able to view the incoming complaints. In the *adminController*, the method complaints execute SQL code that selects and fetches the complaint into the client complaint object list. The list is printed, and the *Admin* can view the complaints. To update the status of a complaint as an administrator, the action method update, checks if the administrator has inserted ”Completed” in the text input that applies to the status attribute in the complaint object. Then the finished date will be added to the complaint and execute the update function in the SQL code. And If the text input is not completed then the status attribute will be updated but without adding a ”Finished date.”

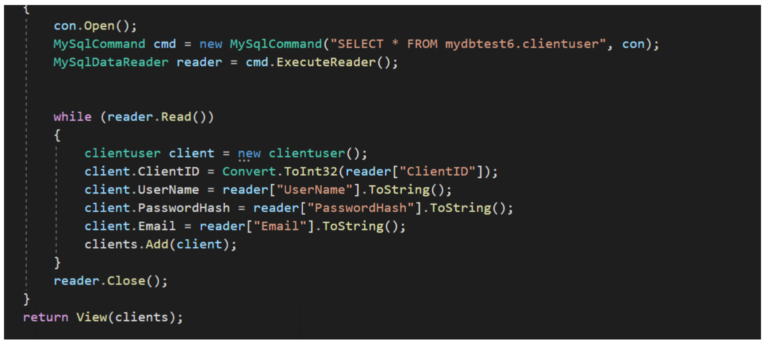


```
using (MySqlConnection con = new MySqlConnection("server=localHost; user=root; database=mydbtest6; port=3306; password=Chidozie
{
    con.Open();
    //---------

    if (complaint.Status.Equals("Completed")) {
        DateTime now = DateTime.Now;

        MySqlCommand cmad = new MySqlCommand("update complaint SET FinishedDate = '" + (now.ToString("F")) + "' where Complaint
        cmad.ExecuteNonQuery();
    }



    //----------


    MySqlCommand cmd = new MySqlCommand("update complaint SET Status = '" + complaint.Status.ToString() + "' where ComplaintID
    cmd.ExecuteNonQuery();
    con.Close();
}
```

Figure 35: implementation of updating profile from the user

4.6.2 User account monitor

For the *Admin* to also know how many accounts are using the complaint management system, an action to read a list of client users from the database was implemented.



```
{
    con.Open();
    MySqlCommand cmd = new MySqlCommand("SELECT * FROM mydbtest6.clientuser", con);
    MySqlDataReader reader = cmd.ExecuteReader();


    while (reader.Read())
    {
        clientuser client = new clientuser();
        client.ClientID = Convert.ToInt32(reader["ClientID"]);
        client.UserName = reader["UserName"].ToString();
        client.PasswordHash = reader["PasswordHash"].ToString();
        client.Email = reader["Email"].ToString();
        clients.Add(client);
    }
    reader.Close();
}
return View(clients);
```

Figure 36: Action method to monitor client users

4.7 Complaint priority

To possibly prioritize different complaints, a priority queue algorithm is necessary. A parameter that is not in the database table is in the model complaint parameter as an integer key value parameter in the model class (see figure 37).

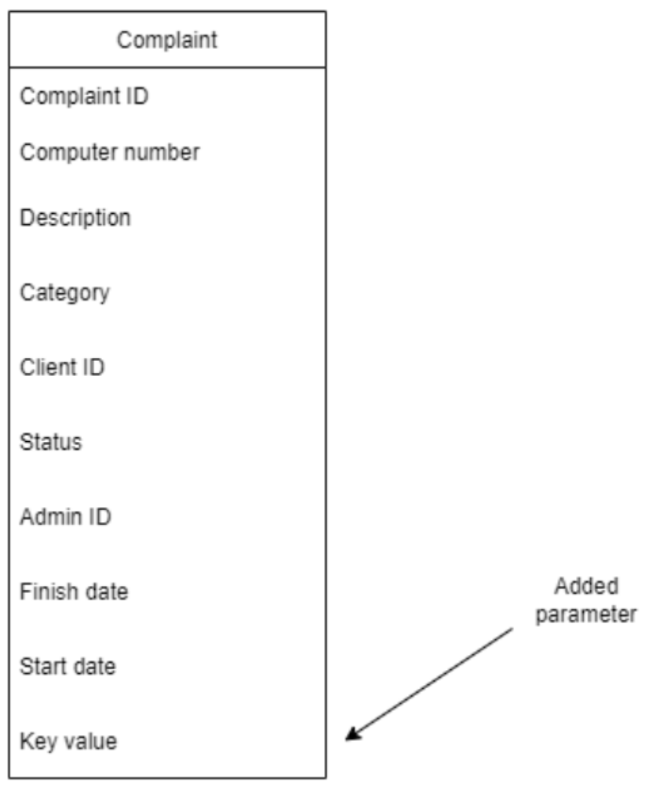


Figure 37: Key value parameter and component

The key value is the integer value that determines the priority depending on the parameter of the same model object complaint. There is only five amount of category that the client can select on the drop-down when sending a complaint. That means there are between 1 and 5. In other words, a higher key value means higher priority. For a key value to be determined, a complaint passes through a method called ”set key value”. ”Set key value method” checks the string value of the complaint category parameter. The key value assigns a value based on the category the client chooses when a complaint is written. The case determines the key value before entering the priority queue.

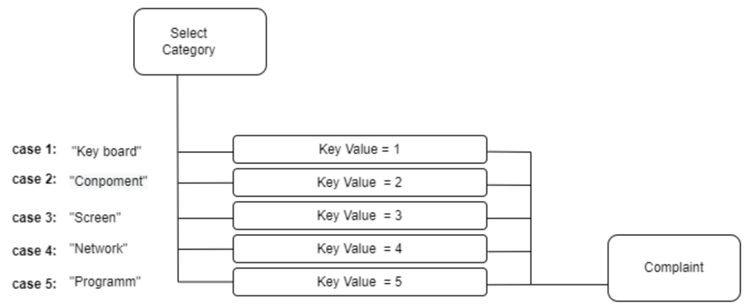
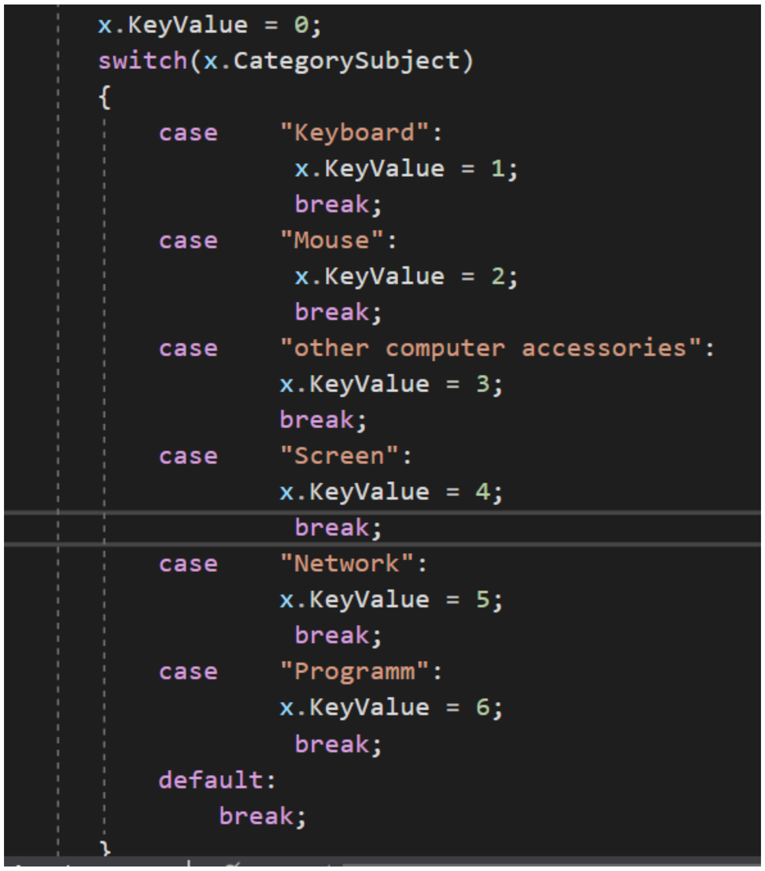


Figure 38: Key value parameter



```
x.KeyValue = 0;
switch(x.CategorySubject)
{
    case    "Keyboard":
             x.KeyValue = 1;
             break;
    case    "Mouse":
             x.KeyValue = 2;
             break;
    case    "other computer accessories":
            x.KeyValue = 3;
            break;
    case    "Screen":
            x.KeyValue = 4;
             break;
    case    "Network":
            x.KeyValue = 5;
             break;
    case    "Programm":
            x.KeyValue = 6;
             break;
    default:
        break;
}
```

Figure 39: Implementation of set key value

4.8 Priority queue and heap structure

Suppose each *key* value ”1,2,3,4,6” in an array or list, the number with the highest *key* value has the highest priority in the queue and, therefore, compliant with *key* value 6 has the highest priority. In contrast, the complaint with the *key* value of 1 has the lowest priority. The queue will be in this order: 6,4,3,2,1 in the list. If, for example, a new complaint with *key* value number 5 enters the list, then the complaint is the second highest. (See figure 40).

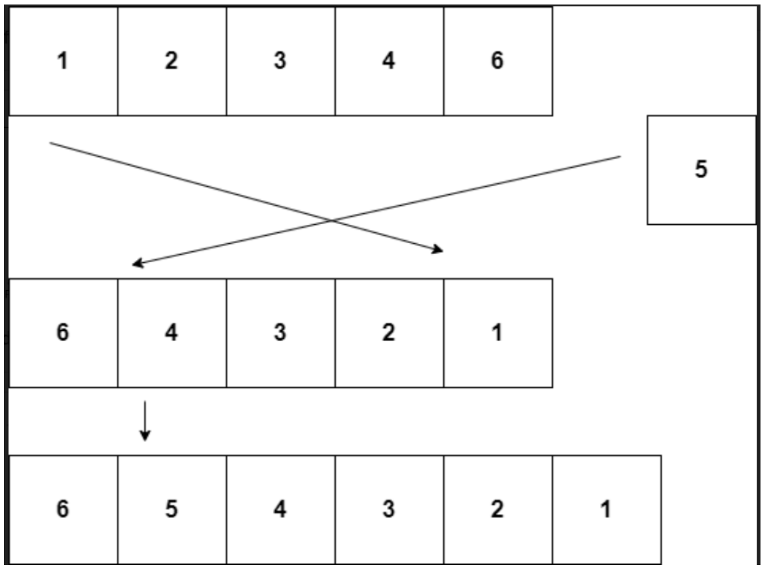


Figure 40: Priority queue for higher value

In order to place the complaints in a list based on the *key* value, a heap structure arranges the priority. A heap is a binary tree structure with several different nodes, whereas the root is at the top of the structure. In the case of the priority queue, the root is the data with the highest priority. A heap structure or a binary consists of nodes, whereas each node has one or two child nodes making it a parent node. The node without a parent node is the root(See Figure 41).

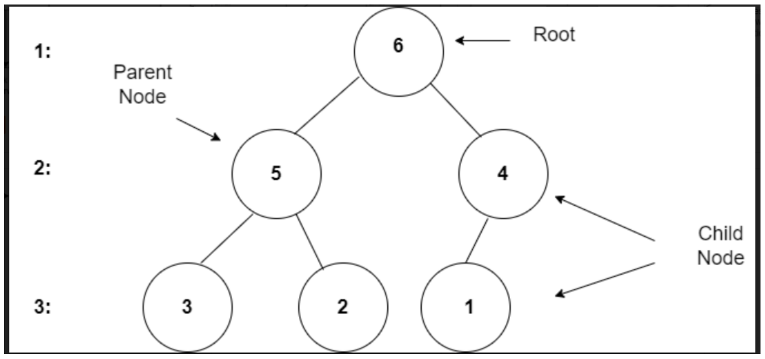
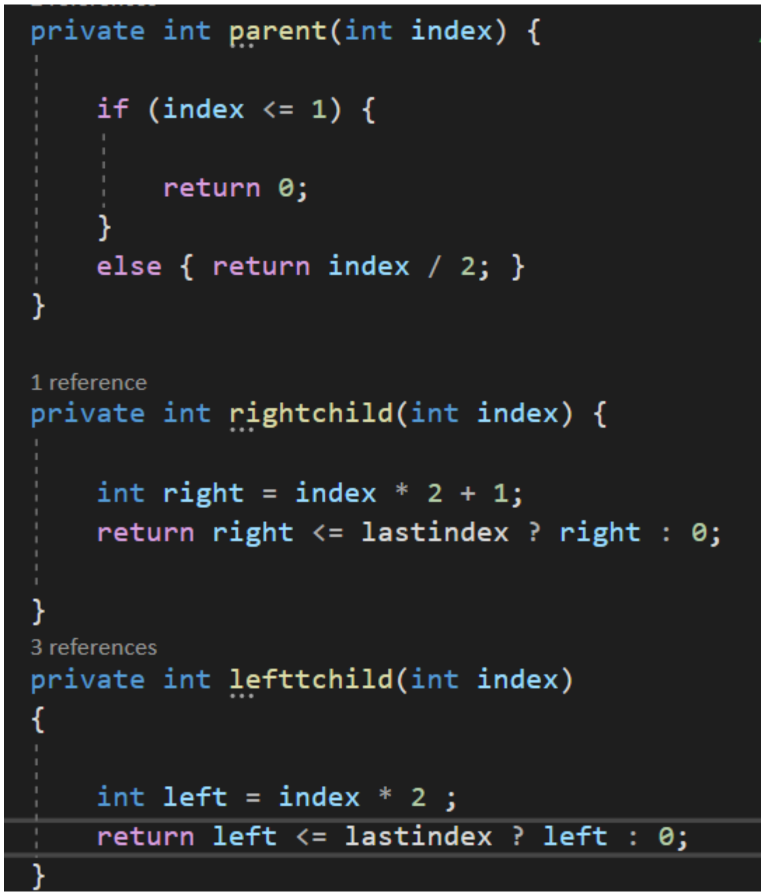


Figure 41: Heap structure of binary tree

In the heap structure, from the first to the last index of the list, it starts from the root as the first index and then the child nodes from the left to the right to the bottom of the structure. A data that enter into a priority queue with heap structure, an array uses a formula $2n+1$ for the left child of the node, $2n+2$ for the right child node, and the parental node with the $(n-1)/2$, for each complaint that enters the list.



```
private int parent(int index) {

    if (index <= 1) {

        return 0;
    }
    else { return index / 2; }
}

1 reference
private int rightchild(int index) {

    int right = index * 2 + 1;
    return right <= lastindex ? right : 0;

}
3 references
private int lefttchild(int index)
{

    int left = index * 2 ;
    return left <= lastindex ? left : 0;
}
```

Figure 42: Parental and child nodes

5 Testing

5.1 Analysis of test result

When the client sends a request, the database checks if a new row in the ”IT support” table is added, which will confirm if the create function is valid. When the status is not validated, the frontend for the IT support is going to confirm and check if the client request is readable in the frontend putting the readable function to the test. The IT support then changes the status as ”validated” on the employee table from the frontend, and to ensure that the status is updated, it will be checked on the database. At the next stage (See figure 43), the IT support will update the status in the ”Employee” table to ”Waiting for confirmation.” When the

”Waiting for confirmation” status is updated by the IT support to ”Completed” in the Employee table, it will be checked in the database that the table is updated. The CRUD function will be monitored by checking the database.

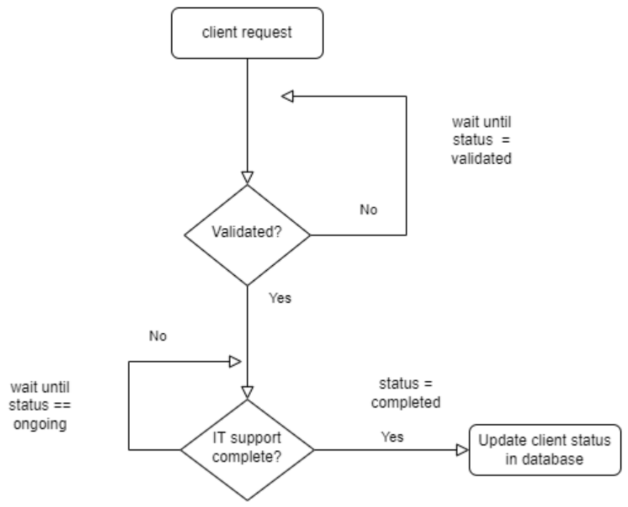


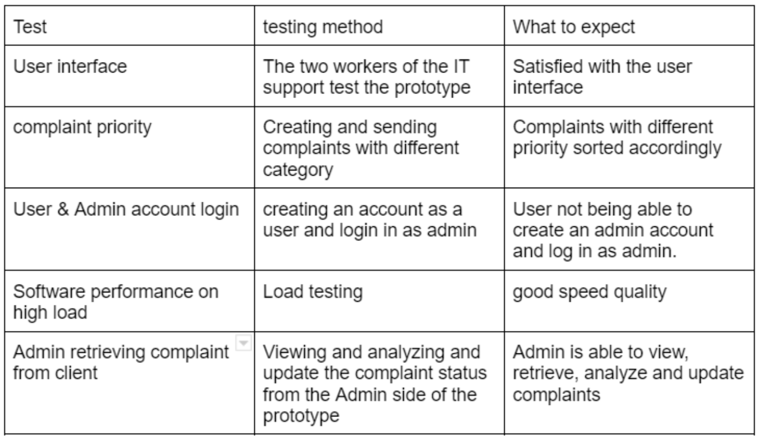
Figure 43: flow chart of the prototype

5.2 Reassurance of the goals and requirement specification

As mentioned in the introduction chapter, the goal of this project is to replace the current way of sending complaints through email with a prototype. The prototype is a complaint management system that is able to receive, send complaints, categorize and prioritize complaints. To identify and confirm the achievement of goals, a usability test, a performance test. Figure 44 is the list of tests to confirm if the goals are reached in this project.

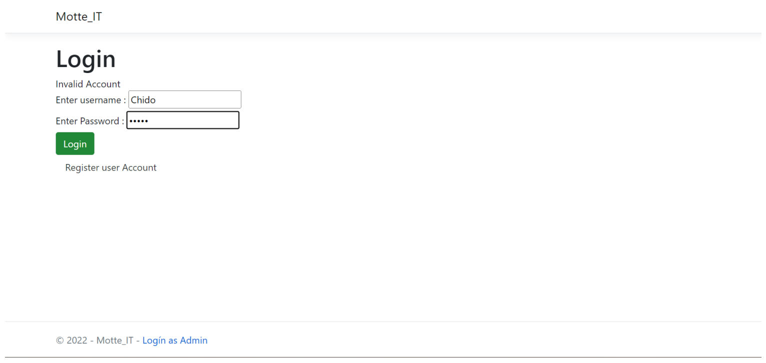
5.3 Testing user account

When a user account does not exist, an error message returns as ”invalid account.”
The failure is due to no account existing in the database.



Test	testing method	What to expect
User interface	The two workers of the IT support test the prototype	Satisfied with the user interface
complaint priority	Creating and sending complaints with different category	Complaints with different priority sorted accordingly
User & Admin account login	creating an account as a user and login in as admin	User not being able to create an admin account and log in as admin.
Software performance on high load	Load testing	good speed quality
Admin retrieving complaint from client	Viewing and analyzing and update the complaint status from the Admin side of the prototype	Admin is able to view, retrieve, analyze and update complaints

Figure 44: table of tests



Motte_IT

Login

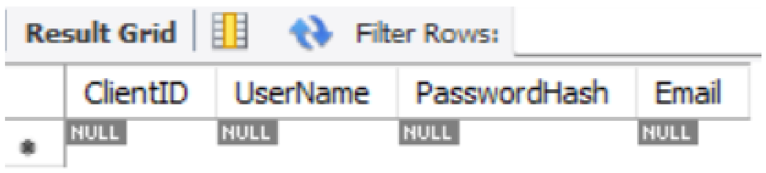
Invalid Account

Enter username : Chido

Enter Password : •••••

© 2022 - Motte_IT - Login as Admin

Figure 45: Login fail

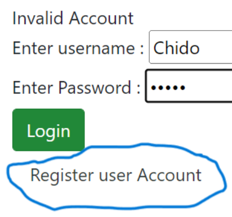


ClientID	UserName	PasswordHash	Email
NULL	NULL	NULL	NULL

Figure 46: Empty data in the clientuser table

The example user registers the account by pressing the ”Register Account” link.

The link navigates to the register page. The user fills in the new account information. After the information is sent, new data in the client table is added, as

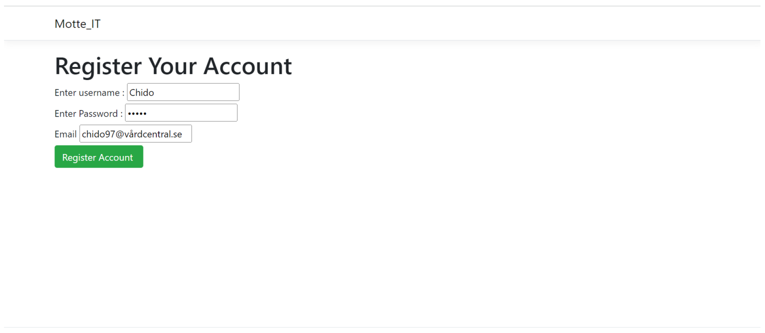

Invalid Account

Enter username : Chido

Enter Password : •••••

Figure 47: Register page

shown in the database.


Motte_IT

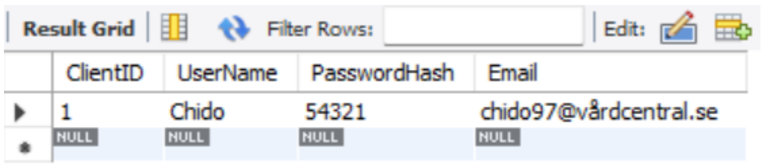
Register Your Account

Enter username : Chido

Enter Password : •••••

Email chido97@vårdcentral.se

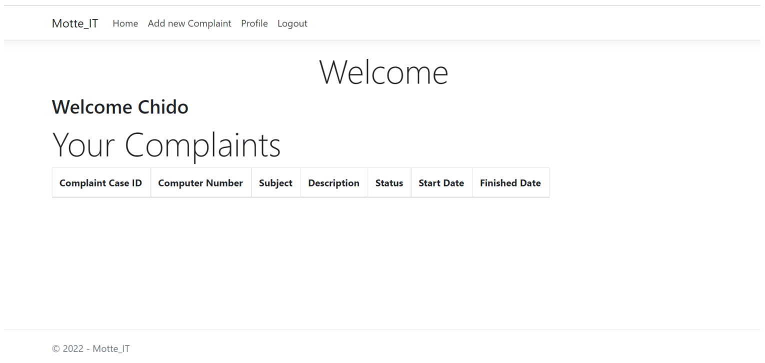
Figure 48: Register page



ClientID	UserName	PasswordHash	Email
1	Chido	54321	chido97@vårdcentral.se
NULL	NULL	NULL	NULL

Figure 49: New client user added to the database

When the user account is registered and added to the database the user can log in. The logged-in user can view the home page and the user name next to the options links of the profile, add complaints and log out into the navigation bar seen in figure 49.



Welcome

Welcome Chido

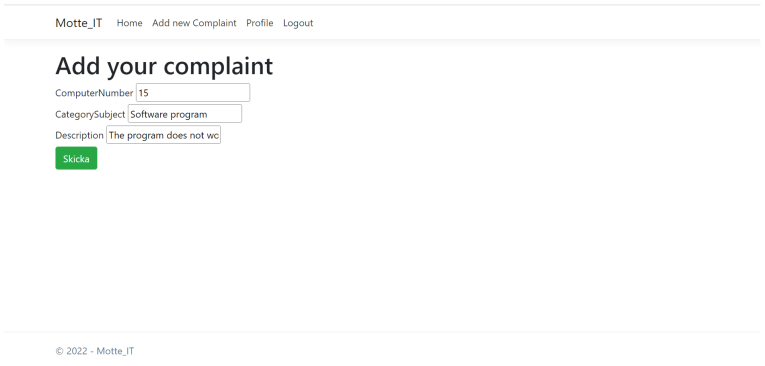
Your Complaints

Complaint Case ID	Computer Number	Subject	Description	Status	Start Date	Finished Date

© 2022 - Motte_IT

Figure 50: Client User page

When the user decides to add a complaint, the ”add complaint” link on the navigation bar navigates the user to another page. The user can write and describe the issue and send it by clicking on the button.



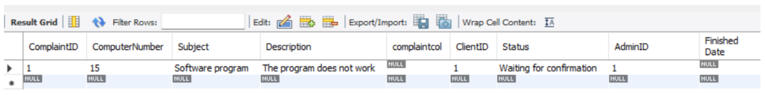
Add your complaint

ComputerNumber	15
CategorySubject	Software program
Description	The program does not wc

© 2022 - Motte_IT

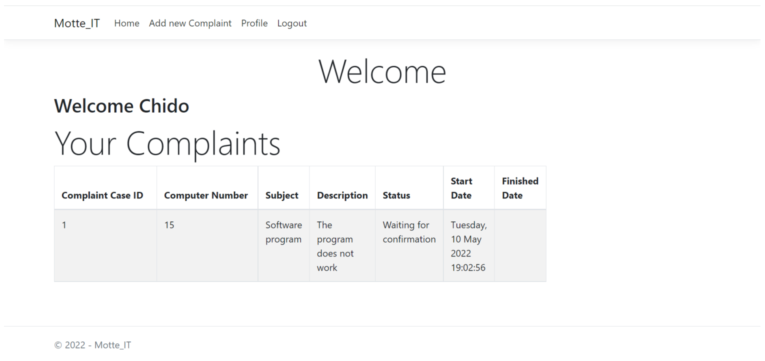
Figure 51: Client User complaint

As the new complaint is sent from the user, a new row in the database is added and is also displayed on the user’s home page to monitor the status of the complaint case



ComplaintID	ComputerNumber	Subject	Description	complaintcol	ClientID	Status	AdminID	Finished Date
1	15	Software program	The program does not work	NULL	1	Waiting for confirmation	1	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 52: Client User complaint



Welcome

Welcome Chido

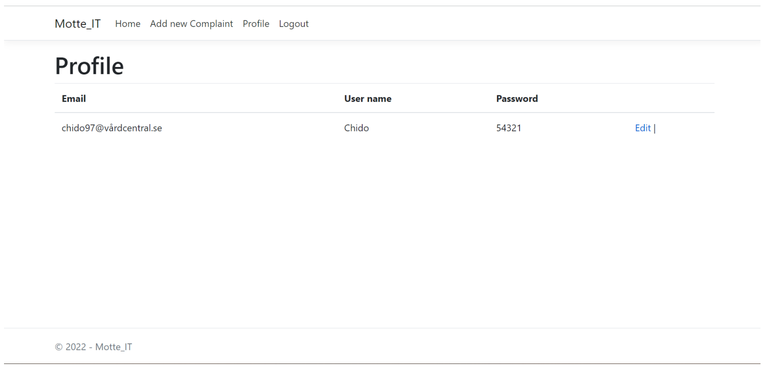
Your Complaints

Complaint Case ID	Computer Number	Subject	Description	Status	Start Date	Finished Date
1	15	Software program	The program does not work	Waiting for confirmation	Tuesday, 10 May 2022 19:02:56	

© 2022 - Motte_IT

Figure 53: Client User complaint

When a user wants to change the user name or password, the user clicks on the navigation link ”Profile”. The user then types the new password and user name. As it is shown in the figure, the data record in the database has been updated.

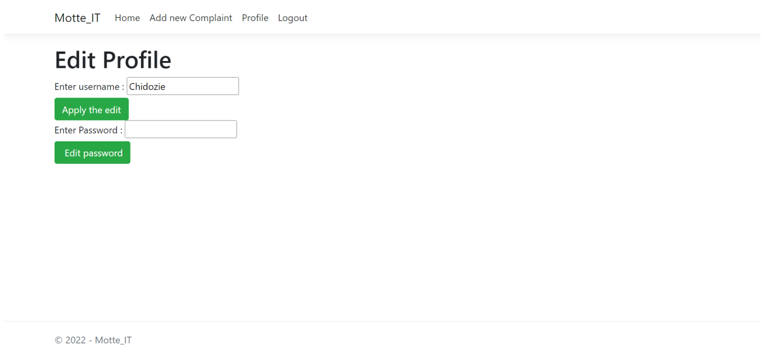


Profile

Email	User name	Password	
chido97@vårdcentral.se	Chido	54321	Edit |

© 2022 - Motte_IT

Figure 54: Client user Profile



Motte_IT Home Add new Complaint Profile Logout

Edit Profile

Enter username : Chidozie

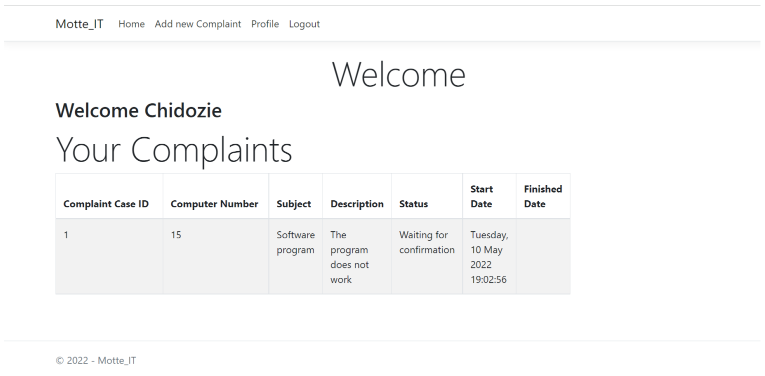
Apply the edit

Enter Password :

Edit password

© 2022 - Motte_IT

Figure 55: Edit profile



Motte_IT Home Add new Complaint Profile Logout

Welcome

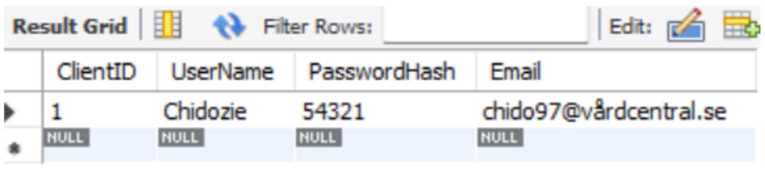
Welcome Chidozie

Your Complaints

Complaint Case ID	Computer Number	Subject	Description	Status	Start Date	Finished Date
1	15	Software program	The program does not work	Waiting for confirmation	Tuesday, 10 May 2022 19:02:56	

© 2022 - Motte_IT

Figure 56: Updated profile Name

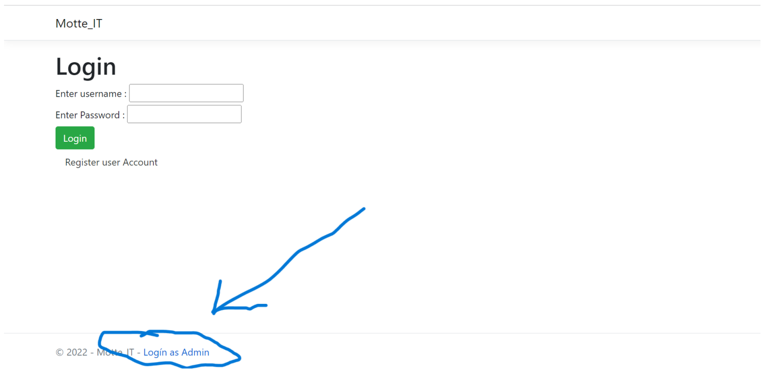


ClientID	UserName	PasswordHash	Email
1	Chidozie	54321	chido97@vårdcentral.se
NULL	NULL	NULL	NULL

Figure 57: Client user name updated in the database

5.4 Analysis of the admin page

To log in as an administrator the IT support clicks on the ”Login as administrator” link. The login input will only work for the *Admin* account. If a client user types in the correct account, it will display an error. When the IT support logs in as administrator, the process will succeed and navigate to the next page.



Motte_IT

Login

Enter username :

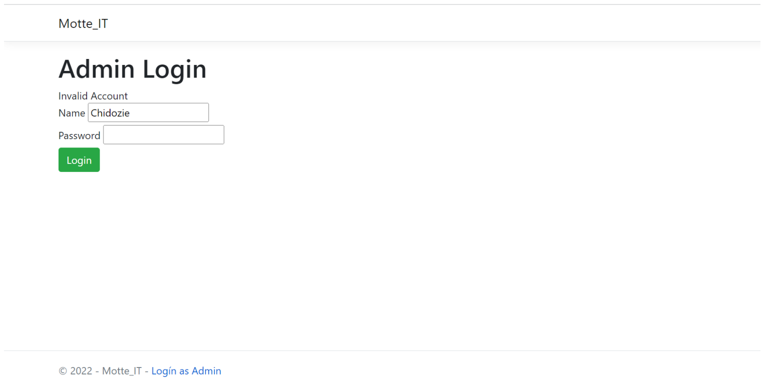
Enter Password :

Login

Register user Account

© 2022 - Motte_IT - Login as Admin

Figure 58: Admin login link



Motte_IT

Admin Login

Invalid Account

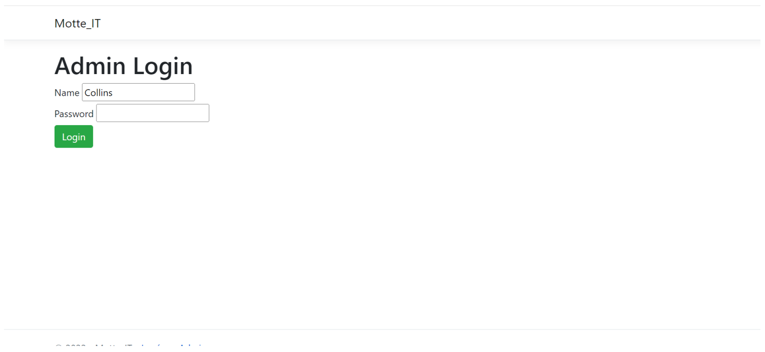
Name Chidozie

Password

Login

© 2022 - Motte_IT - Login as Admin

Figure 59: User login in at Admin login page


Motte_IT

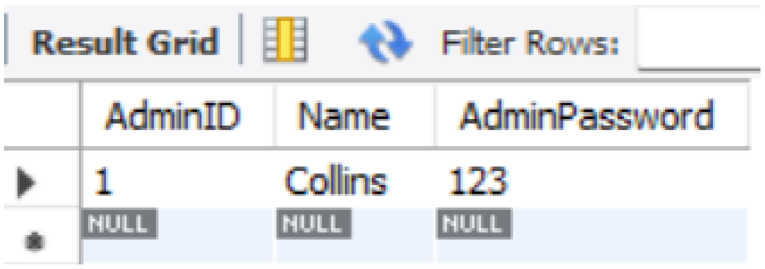
Admin Login

Name Collins

Password

Login

Figure 60: The administrator login in



AdminID	Name	AdminPassword
1	Collins	123
NULL	NULL	NULL

Figure 61: The administrator database record

The admin page consists of a navigation bar with a link to a page of complaints from the client user, and a link to a page with a list of users.

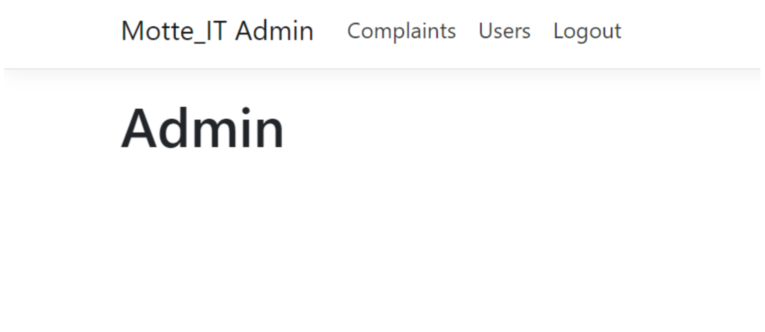
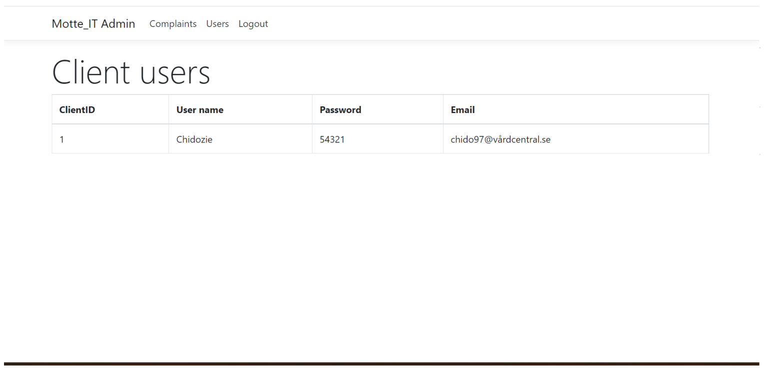


Figure 62: Admin Home Page

The list of the complaints is shown and sent from the different client users. The same information can be shown in the database. The example complaint seen in the list is validated by IT support. The validation is shown on the user example page.



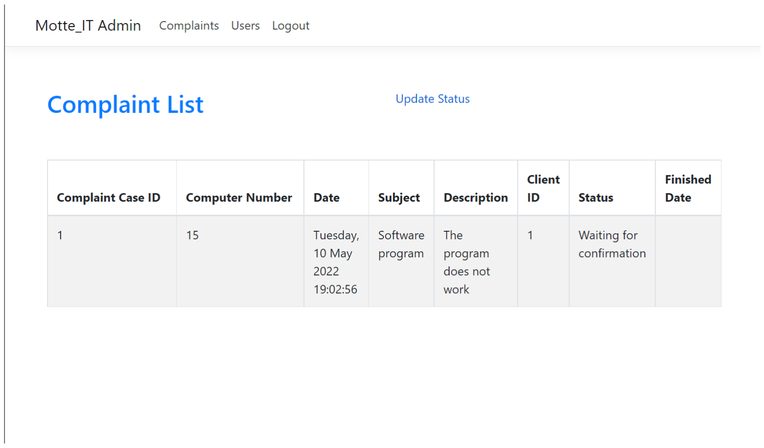
Motte_IT Admin Complaints Users Logout

Client users

ClientID	User name	Password	Email
1	Chidozie	54321	chido97@vårdcentral.se

Figure 63: Client user from the admin's side

When the task of fixing the issue sent by the client user is completed, The IT administrator updates the complaint as the task is complete. It is also shown on the example user page.



Motte_IT Admin Complaints Users Logout

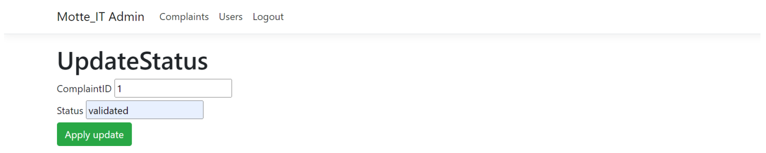
Complaint List

Update Status

Complaint Case ID	Computer Number	Date	Subject	Description	Client ID	Status	Finished Date
1	15	Tuesday, 10 May 2022 19:02:56	Software program	The program does not work	1	Waiting for confirmation	

Figure 64: Admin complaints

When the task is complete, the complaint case will also update the finished date.

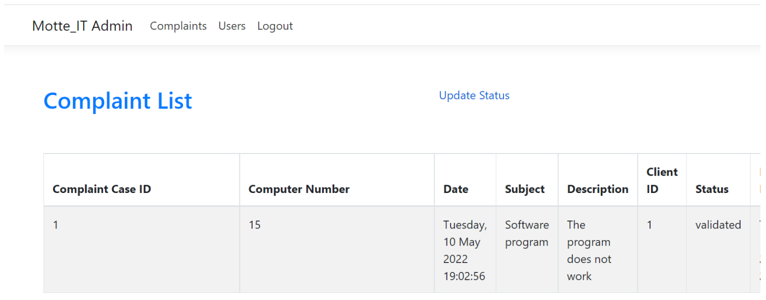

Motte_IT Admin Complaints Users Logout

UpdateStatus

ComplaintID 1

Status validated

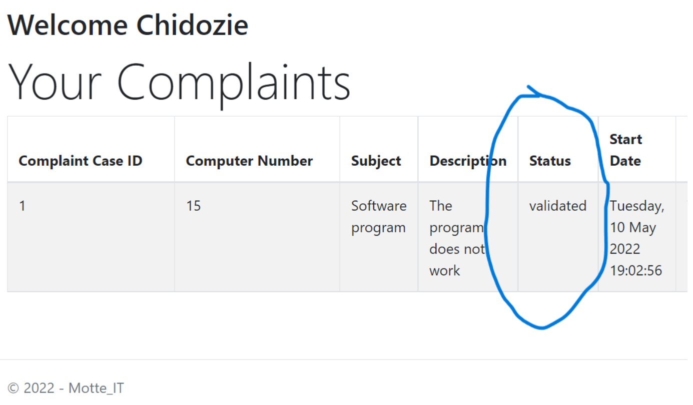
Figure 65: Admin updating status


Motte_IT Admin Complaints Users Logout

Complaint List

Complaint Case ID	Computer Number	Date	Subject	Description	Client ID	Status
1	15	Tuesday, 10 May 2022 19:02:56	Software program	The program does not work	1	validated

Figure 66: Complaint validated by the IT support admin

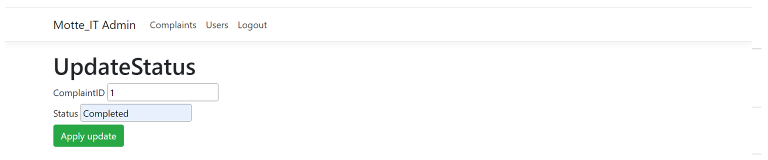

Welcome Chidozie

Your Complaints

Complaint Case ID	Computer Number	Subject	Description	Status	Start Date
1	15	Software program	The program does not work	validated	Tuesday, 10 May 2022 19:02:56

© 2022 - Motte_IT

Figure 67: Complaint validated by the IT support admin from the user side

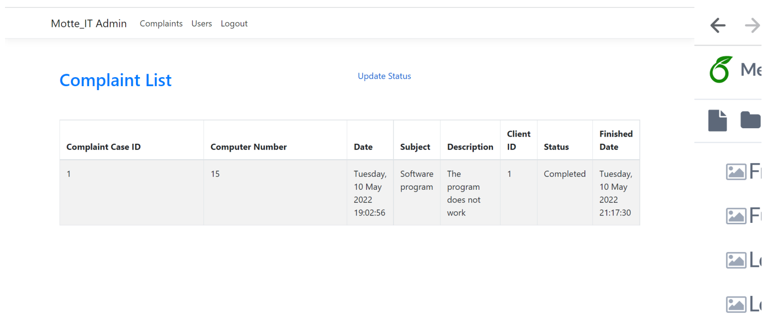


UpdateStatus

ComplaintID 1

Status Completed

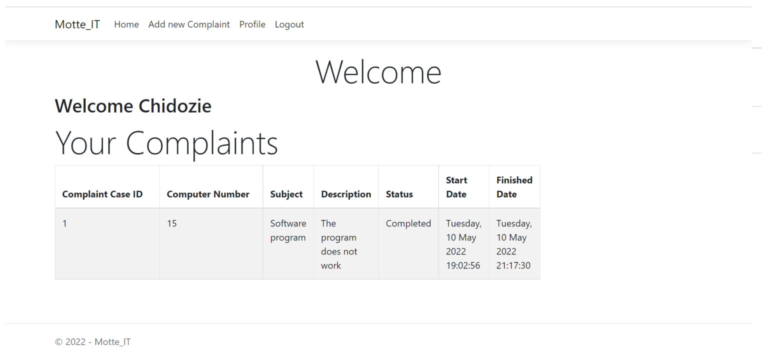
Figure 68: The complaint validated by the IT support admin



Complaint List

Complaint Case ID	Computer Number	Date	Subject	Description	Client ID	Status	Finished Date
1	15	Tuesday, 10 May 2022 19:02:56	Software program	The program does not work	1	Completed	Tuesday, 10 May 2022 21:17:30

Figure 69: Complaint completed and updated by the IT support admin



Welcome

Welcome Chidozie

Your Complaints

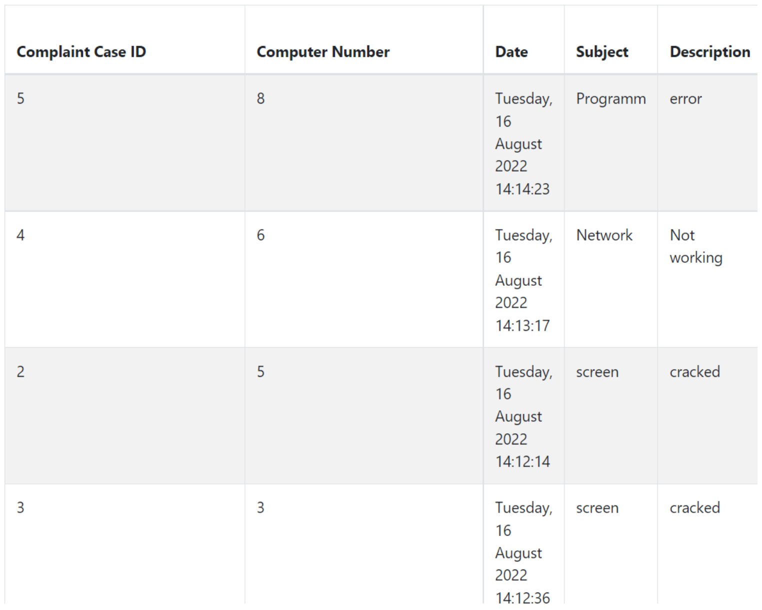
Complaint Case ID	Computer Number	Subject	Description	Status	Start Date	Finished Date
1	15	Software program	The program does not work	Completed	Tuesday, 10 May 2022 19:02:56	Tuesday, 10 May 2022 21:17:30

© 2022 - Motte_IT

Figure 70: Complaint completed by the IT support admin from the client user side

5.5 Priority Complaints

Figure 71 showcases the result of complaint priority in the list that is sorted based on the category (subject) in ascending order.

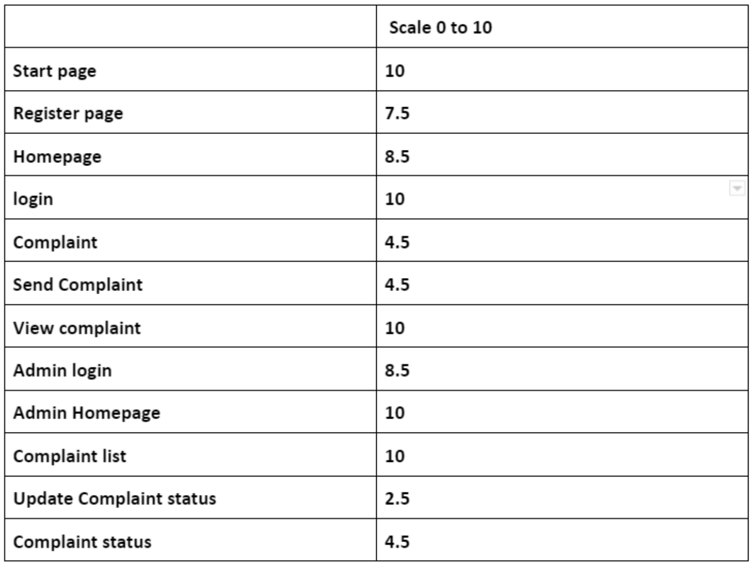


Complaint Case ID	Computer Number	Date	Subject	Description
5	8	Tuesday, 16 August 2022 14:14:23	Programm	error
4	6	Tuesday, 16 August 2022 14:13:17	Network	Not working
2	5	Tuesday, 16 August 2022 14:12:14	screen	cracked
3	3	Tuesday, 16 August 2022 14:12:36	screen	cracked

Figure 71: Priority tests

5.6 User interface test

The test showcases each part of the complaint management prototype rating (See figure 72).



	Scale 0 to 10
Start page	10
Register page	7.5
Homepage	8.5
login	10
Complaint	4.5
Send Complaint	4.5
View complaint	10
Admin login	8.5
Admin Homepage	10
Complaint list	10
Update Complaint status	2.5
Complaint status	4.5

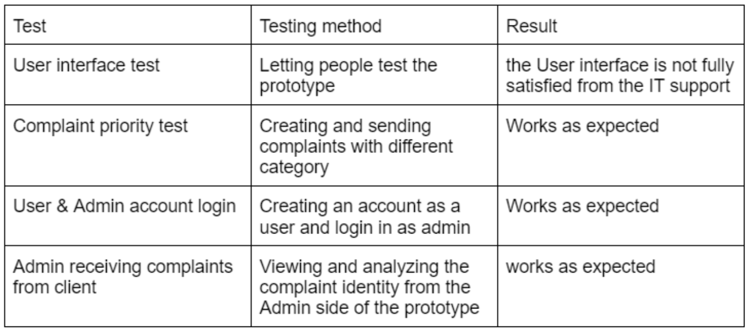
Figure 72: Rates of each section

$$7.16 = \frac{1}{12}\sum_{i=1}^{12} x_i = \frac{1}{12}\left(10_1 + 7.5_2 + 8.5_3 + 10_4 + 4.5_5 + 4.5_6 + 10_7 + 8.5_8 + 10_9 + 10_{10} + 2.5_{11} + 4.5_{12}\right)$$

The overall result showcases the user satisfaction to score 7.16 out of 10, which gives us a rating of 71.6 percent.

5.7 Test summary

Figure 73 presents the overall project results.



Test	Testing method	Result
User interface test	Letting people test the prototype	the User interface is not fully satisfied from the IT support
Complaint priority test	Creating and sending complaints with different category	Works as expected
User & Admin account login	Creating an account as a user and login in as admin	Works as expected
Admin receiving complaints from client	Viewing and analyzing the complaint identity from the Admin side of the prototype	works as expected

Figure 73: Project results

6 Discussion

6.1 Test results

Four different tests were made for the complaint management system prototype in this project.

Based on the requirements, The first test was for two IT support workers to rate 12 different web pages of the complaint management system as user experience on a scale of 0 to 10. The aim was to get the rating at 100 percent to imply that the user interface is fulfilled. The overall user interface resulted in 71.16 percent. This implied the IT support workers were not fully satisfied with the result.

The second test resulted in the complaints having different priorities. The category or subject with higher priorities that are sent sorts in the order based on the complaint with the category that has the highest priority.

Third test result works as expected. Only the *Admin* could log in as an administrator. Any user that registers itself cannot log in on the admin side.

Fourth test resulted in complaints that are being sent from the user, or the so-called client side can be viewed on the admin side. The result also shows the status of the complaint being updated. As the status is updated, the client can view it from their login account.

6.2 The projects reconciliation

The first requirement is to make the complaint management system a good user interface for IT support. The user interface test's overall rating is 71.16 percent in total, which means it does not fully fulfill the first requirement. Seven out of 12 pages have more room to improve.

The second requirement is to give the complaints different priorities depending on the category that the user selects. The results of the test for complaint priority fulfill the requirement.

The third requirement is the *Admin* receiving complaints being sent from the clients. The testing result shows cases where the information can be successfully updated and analyzed. As the user sends the complaint information, the testing shows the information is stored in the database. On the Admin page, the IT support can retrieve the information from the admin login side.

The fourth requirement is Admin login. The testing shows that when the user registers its information, the data is sent and retrieved in the database. As the user tries to login into the admin side it will not work. Therefore the fourth requirement is fulfilled

6.3 The evaluation of Requirement Specification

Regarding the user interface test, seven pages out of twelve felt incomplete for IT support. For instance, when the user wants to select a category for a complaint, The user needs to type the category. The issue is that if the user types only one wrong character, it will not work because, looking at figure 39, the string character needs to be exact with the letter the user types from the user interface. Another example is when the *Admin* needs to update the status. The update status page felt misplaced and was not necessary. An edit bottom next to each of the complaints could be a better option.

Having a list that prioritizes different complaints, makes the IT support experience it to be more efficient.

The two IT support workers from the HCCH tested the prototype of the complaint management system and experienced the testing of the prototype as ”simple”. The process of sending in a complaint does not require many steps. The IT support feedback was positive. Experience in administrating the complaint management system made it easier to respond and confirm the issue. The IT support experience it to more efficiently keep track of the different cases sent from the client user, and manage all the incoming information is one of the goals for the prototype in this project.

6.4 Comparison to the related works

For one of the related work in related work section 2.5.3 Help desk, the outcome of the result was a similar result. The difference was that the complaint management system includes a phone texting function. The phone message function could be an improvement for the complaint management system project in the HCCH due to sending instant text messages as a notification. Receiving notifications when the client user is not using the complaint management system could be efficient for the working environment. In this current project, there are no notifications that can be sent when the status of the complaint has been updated.

6.5 Result of economic and environmental perspective

The expense of the complaint management depends on the computer power of the server that is required. If the IT support were to expand the complaint management system to other major clients such as schools and other healthcare centers, The cost would increase due to higher numbers of users. For the server to keep up with the higher numbers of users, it would need more CPU (central processing unit), RAM (Random access memory), or SSID (Service Set Identifier). Expanding the complaint management system also has its effect from the environmental point of view. Since the complaint management system is powered by the server when there is more computational power needed, more computer chips are ordered. It causes more production of computer chips which affects the environment due to pollution.

6.5.1 Integrity and security

The complaint management system requires registration from any new user that wants to send a complaint. The information that is registered from the user is not personal or private, only the user name, password, and email. The information that is relevant for IT support is an issue with the healthcare hjärtat computers from whom it was sent, not any personal information. The prototype of the complaint management system creates data integrity, where it only retrieves no information other than the complaint. Storing none sensitive information makes the complaint management system secure if there were to occur a database attack.

6.5.2 Social

From the social aspect, direct communication between the worker and the IT support will decrease. When the client sends a complaint through email, the communication is more direct. With the complaint management system prototype, the IT support only responds by updating the status of the complaint case.

6.6 Room for future improvement

Due to a lack of experience in this project and time limitations, the user interface was not fully complete with quality. A way to improve the user interface is to implement a drop-down list for the user to select a category instead of risking miss typing. The list of complaints on the admin side can be improved by adding an edit button in each row to update the status instead of entering the next page to type in the status of the complaint. To get an optimal test result, an increase in test participants will make the test result even more accurate.

When the *Admin* updates a complaint status, the complaint remains in the same list of the other complaints that are waiting for an update. A way to improve the priority list is to separate complaints into different lists. That way is easier for the *Admin* to keep track of all the complaints.

The complaint management system lacks user confirmation from the admin. A none eligible user that has nothing to do with the complaint management system can register and spam the system. To improve and ensure that such a case will not occur, as a user registers itself, the *Admin* identifies the user's employee number before confirming.

Even if the complaint management system does not store personal information, a way to make the system have more integrity to encrypt the user password. Then the *Admin* does not know the user password. If there were to be a security breach by hackers, all the passwords in the database would be encrypted.

Another feature that improves the complaint management prototype is sending notifications to the users when the complaint case is being updated and completed by IT support. In that way, the workers in the HCCH will know for sure the computer is fixed and can be used again, which makes it more time efficient.

6.7 Deployment and maintenance

The remaining waterfall methodology after the test phase is deployment and maintenance. The goal of this project is to replace the current way of sending complaints through email with a prototype of the complaint management system. Deployment and maintenance phases needed to happen for the project goal to be fully achieved. This project did execute the deployment and maintenance phase due to the result of the test not fully achieving the goals. The prototype only tests the functionality, but with the deployment and maintenance phase, more test results can be presented and analyzed by putting the prototype of the complaint management system into more practice, like testing the system on new major future clients. The maintenance would also check the system would handle the load and the effect of response time. In that case, the result would showcase a result of the scalability of the prototype. The goal of this project was to analyze the scaling. Without the maintenance process, present the result of scalability is not achievable.

7 Conclusion

IT support of HCCH today is using the emailing system to manage complaints when an issue occurs on the computer. IT support is planning to expand its services to other major clients such as schools and other healthcare centers. Using email as a complaint would not be efficient due to managing and keeping track of complaints. A prototype of a complaint management system is implemented in this project. The methodology of this project was the waterfall method, starting with planning, designing with MVC software architect pattern, and Developing and testing. The result of the project partly fulfilled the goals and has more room for improvements in the future.

References

[1] Agrawal R., Nyamful C. Challenges of big data storage and management. Global Journal of Information Technology: Emerging Technologies. 2016;6(1):1-10.

[2] Adebisi O., Oladosu D., Busari O., Oyewola Y. Design and implementation of hospital management system. International Journal of Engineering and Innovative Technology (IJEIT). 2015;5(1).

[3] Nasr O., Alkhider E. Online complaint management system. International Journal of Innovative Science, Engineering & Technology. 2015.

[4] Elsweiler D., Baillie M., Ruthven I. What makes re-finding information difficult? A study of email re-finding. In: European Conference on Information Retrieval. Springer; 2011. pp. 568-79.

[5] Balter O. Keystroke level analysis of email message organization. In: Pro- ¨ ceedings of the SIGCHI conference on Human factors in computing systems; 2000. pp. 105-12.

[6] Ducheneaut N., Bellotti V. E-mail as habitat: an exploration of embedded personal information management. interactions. 2001;8(5):30-8.

[7] Nguyen Nhat M. Building a component-based modern web application: fullstack solution. 2018.

[8] Held G. Server management. CRC Press; 2000.

[9] Sharma V., Dave M. Sql and nosql databases. International Journal of Advanced Research in Computer Science and Software Engineering. 2012;2(8).

[10] Truica CO., Radulescu F., Boicea A., Bucur I. Performance evaluation for CRUD operations in asynchronously replicated document oriented database. In: 2015 20th International Conference on Control Systems and Computer Science. IEEE; 2015. pp. 191-6.

[11] Picken JC. From startup to scalable enterprise: Laying the foundation. Business Horizons. 2017;60(5):587-95.

[12] Hardianto H., Shofi IM., Khairani D., Subchi I., Ginanto DE., Hidayati A. Integration of the Helpdesk System with Messaging Service: A Case Study Approach. In: 2021 9th International Conference on Cyber and IT Service Management (CITSM). IEEE; 2021. pp. 1-5.

[13] Rachmawati E., Kom M., Kom M. Web-Based Ticketing System Helpdesk Application Using CodeIgniter Framework (Case Study: PT Commonwealth Life). International Journal of Computer Science and Mobile Computing. 2018;7(12):29-41.

[14] Virata AJA., Sergio NA. Impact of IT Job-order e-Ticketing System on Enduser Satisfaction: An Empirical Analysis. SDCA Asia-Pacific Multidisciplinary Research Journal Volume 2, December 2020:4.

[15] Alex AOI., Uzoamaka EO. Design and Implementation of a Tertiary Institution Web-Based Student Complaint Management System.

[16] Al-Sharji S., Al-Mahruqi A., Kumar R. Help Desk Management System for PC Troubleshooting. 2014.

[17] Crespo-Santiago CA., Cosme SdlCD. Waterfall method: a necessary tool for implementing library projects. HETS Online Journal. 2011;1(2):86-99.

[18] Boiano S., Bowen JP., Gaia G. Usability, design and content issues of mobile apps for cultural heritage promotion: The Malta culture guide experience. arXiv preprint arXiv:12073422. 2012.

[19] Sahu A. 9 advantages of mobile apps over responsive eCommerce websites; 2022. Available from: https://www.westagilelabs.com/blog/9-advantagesof-mobile-apps-over-responsive-ecommerce-websites/.

[20] Masresha T. Changing Desktop application to real time Web application. 2018.

[21] Pop P. Comparing web applications with desktop applications: An empirical study. Linköping University, Linköping. 2002.

[22] Adhikari A. Full stack javascript: Web application development with mean. 2016.

[23] Yoon IC., Sussman A., Memon A., Porter A. Effective and scalable software compatibility testing. In: Proceedings of the 2008 international symposium on Software testing and analysis; 2008. pp. 63-74.

[24] Del Sole A. Visual Studio 2015 Succinctly. Morrisville: Syncfusion; 2014. [25] Doglio F. Pro REST API Development with Node. js. Apress; 2015.

[26] Chitra LP., Satapathy R. Performance comparison and evaluation of Node. js and traditional web server (IIS). In: 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET). IEEE; 2017. pp. 1-4.

[27] Ravago JAF. Comparison of MySQL and MS SQL Server.

[28] Deari R., Zenuni X, Ajdari J., Ismaili F., Raufi B. Analysis and comparison of document-based databases with sql relational databases: Mongodb vs mysql. In: Proceedings of the International Conference on Information Technologies (InfoTech 2018); 2018. pp. 1-10.

[29] Chaqfeh M, Haseeb M, Hashmi W, Inshuti P, Ramesh M, Varvello M, et al. To Block or Not to Block: Accelerating Mobile Web Pages On-The-Fly Through JavaScript Classification. arXiv preprint arXiv:210613764. 2021.