**Challenges:**

* **Support many languages**

Our project support more than one programming language. We have two main languages Java and C++. In the future work. Our system will support many popular languages like a JavaScript, Python, SQL, and PHP….etc. We will talk about our two main languages:

1. **JAVA programming language:**

Java is a programming language and computing platform first released by Sun Microsystem in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. Like any programming language, the Java language has its own structure, syntax rules, and programming paradigm. The Java language’s programming paradigm is based on the concept of OOP, which the language’s features support. Once a program has been written, the high-level instructions are translated into numeric codes that computers can understand and execute.

Java was designed with a few key principles in mind:

* **Easy to Use:** The fundamentals of Java came from a programming language called C++. Although c++ is powerful language it was felt to be too complex in its syntax, and inadequate for all of Java's requirements. Java built on, and improved the ideas of c++, to provide a programming language that was powerful and simple to use.
* **Reliability:** Java needed to reduce the likelihood of fatal errors from programmer mistakes. With this in mind, object-oriented programming was introduced. Once data and its manipulation were packaged in one place, it increased java's robustness.
* **Secure:** As Java was originally targeting mobile devices that would be exchanging data over networks, it was built to include a high level of security. Java probably the most secure programming language to date.
* **Platform Independent:** programs needed to work regardless of the machine they were being executed on. Java was written to be a portable language that doesn’t care about the operating system (OS) or the hardware of the computer.

Our project is fully developed by Java programming language. Also, support judging on Java language.

1. **C++ programming language:**

C++ is a general-purpose object-oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of the C language. It is therefore possible to code C++ in a "C style" or "object-oriented style." In certain scenarios, it can be coded in either way and is thus an effective example of a hybrid language. C++ is considered to be an intermediate-level language, as it encapsulates both high- and low-level language features. Initially, the language was called "C with classes" as it had all the properties of the C language with an additional concept of "classes." However, it was renamed C++ in 1983.

C++ is one of the most popular languages primarily utilized with system/application software, drivers, client-server applications and embedded firmware.

C++ was designed with a few key principles in mind:

* **Nature of Language**: C++ is an object-oriented programming language and supports Polymorphism, Abstract Data Types, Encapsulation, among others. Even though C++ derives basic syntax from C, it cannot be classified as a structural or a procedural language.
* **Point Of Emphasis:** C++ emphasizes the objects and not the steps or procedures. It has higher abstraction level.
* **Compatibility with Overloading**: C++ supports function overloading, implying that one can have name of functions with varying parameters.
* **Compatibility with Exception Handling**: C++ supports Exception Handling can be done through try & catch block.

And other many principles. Now we will explain the language we used to build our database.

1. **SQLite:**

SQLite is an in-process library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite doesn't have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers and views, is contained in a single disk file. The database file format is cross platform – you can freely copy a database between 32-bit and 64-bit systems or between big and little endian architectures. These features make SQLite a popular choice as an Application File Format.

* **Synchronization between threads and servers (network challenge):**

In our project threads are processes made by users, we have to synchronize that threads over multiple servers. Firstly we have to talk about Synchronization, Threads, and Servers.

* **Synchronization**: Synchronization, in the context of.NET, is a process that involves coordinating the execution of multiple threads to ensure a desired outcome without corrupting the shared data and preventing any occurrence of deadlocks and race conditions.
* **Thread:** A thread is a path of execution within a process. A process can contain multiple threads.
* **Server:** A server is a computer [program](https://searchsoftwarequality.techtarget.com/definition/program) or device that provides a service to another computer program and its user, also known as the client. In a data center, the physical computer that a server program runs on is also frequently referred to as a server. That machine may be a [dedicated server](https://searchmicroservices.techtarget.com/definition/dedicated-server) or it may be used for other purposes as well.

We will explain how we achieve that concept in our project.

* **Management of 3 servers:**

**Server management can be defined as the tasks and services that are done on a server in order to manage it.** This usually entails:

* **Monitoring** of the server and apps running on the server. Checking their status, uptime, and monitoring for any new or recurring issues.
* **Updating** the server and software installed on the server. Although most server management companies offer this as part of their services, some still don’t. Nevertheless, it’s still considered as part of the ‘server management’ process.
* **Setup** and configuration. The actual server setup and configuration of software and services running on the server. Again, this may not be a part of the server management plan offered by some companies, but it most often is.

Now we will explain why we need server management.

* **Fewer costs:** Instead of hiring a full-time system administrator (which can be quite costly), you can get a server management plan and save hundreds (if not thousands) of dollars on a monthly basis. Some server providers offer a ‘managed service’ as an add-on to their servers, but it often is much more expensive than getting managed services from a 3-rd party provider.
* **Quick** **turnaround time**: Depending on the provider itself and the experience of the sysadmins involved, the turnaround time for server issues and tasks is fairly quick. So instead of wasting hours on troubleshooting and research, you can have experts work on your server and fix any issues in a matter of minutes. This still depends on the severity and complexity of the issue itself, but when compared, having expert server managers working on your server is far quicker than doing it yourself.
* **Fewer worries:** No need to stay updated on the latest security patches and releases, your server will be taken care of.

We will explain how we achieve that concept in our project.

* **Reliability:**

Reliability means that project stay reliable while number of users increases, we will discuss Reliability in our project.

* **Good performance and great GUI (UI/UX):**

We use JavaFx GUI in our project. JavaFX is designed to provide applications with such sophisticated GUI features as smooth animation, web views, audio and video playback, and styles based on Cascading Style Sheets (CSS). GUI helps any user to understand the system also how to interact with it.

**Benefits of JavaFx GUI:**

* You can write JavaFX Swing programs using much less code because JavaFX performs all of the Swing-based housekeeping for you. You don't have to register event listeners and it writing action functions is more concise. Furthermore, with the JavaFX binding mechanism, it is simple to integrate the GUI components with your underlying model. Binding in JavaFX is very powerful.

Some screenshots of GUI in our system.

* The JavaFX declarative style not only makes your code more readable, but it makes it more concise too.
* While you can use the JavaFX swing components, you can also use JavaFX "native" components that include Textbox, Button, Hyperlink, Toggle Button, Radio Button, Toggle Group, Checkbox, List View, Label, Scrollbar, Slider, Progress-Bar, and Progress Indicator. The advantage of the "native" controls is that they work in all environments: desktop, mobile, while swing-based controls work only in the desktop environment.
* **Using FXML**

FXML is an XML-based markup language that enables developers to create a user interface (UI) in a JavaFX application separately from implementing the application logic. Swing has never offered a declarative approach to building a user interface. The declarative method for creating a UI is particularly suitable for the scene graph, because the scene graph is more transparent in FXML. Using FXML enables developers to more easily maintain complex user interfaces.

* **CSS Support**

Cascading style sheets contain style definitions that control the look of UI elements. The usage of CSS in JavaFX applications is similar to the usage of CSS in HTML. With CSS, you can easily customize and develop themes for JavaFX controls and scene graph objects. Using CSS as opposed to setting inline styles enables you to separate the logic of the application from setting its visual appearance. Using CSS also simplifies further maintenance of how your application looks and provides some performance benefits.

**Problems**:

We have some types of problems. We will explain each one separately.

* **Network problems:**

1. **Bottleneck:** A network bottleneck refers to a discrete condition in which data flow is limited by computer or network resources. The flow of data is controlled according to the bandwidth of various system resources. If the system working on a network is delivering a higher volume of data than what is supported by the existing capacity of the network, then a network bottleneck will occur. As the name implies, a network bottleneck results in slow communication speeds and limits user efficiency and productivity on a network. To avoid all these problems, systems are built to support a particular data flow capacity so that work can continue without any issues. On a network, each system is able to work according its processor speed, its memory size, its cache speed and its network interface card speed. These discrete systems do not rely on other network resources to accept their incoming data at the rate they are sending because these objects only receive data according to their own capacity.
2. **Heavy Traffic:** Network traffic refers to the amount of data moving across a network at a given point of time. Network data is mostly encapsulated in network packets, which provide the load in the network. Network traffic is the main component for network traffic measurement, network traffic control and simulation. The proper organization of network traffic helps in ensuring the quality of service in a given network.

Network traffic can be broadly classified into the following categories:

* Busy/heavy traffic - High bandwidth is consumed in this traffic
* Non-real-time traffic - Consumption of bandwidth during working hours
* Interactive traffic - Is subject to competition for bandwidth and could result in poor response times if prioritization of applications and traffic is not set
* Latency-sensitive traffic - Is subject to competition for bandwidth and could result in poor response times

Proper analysis of network traffic provides the organization with the following benefits:

* Identifying network bottlenecks - There could be users or applications that consume high amounts of bandwidth, thus constituting a major part of the network traffic. Different solutions can be implemented to tackle these.
* Network security - Unusual amount of traffic in a network is a possible sign of an attack. Network traffic reports provide valuable insights into preventing such attacks.
* Network engineering - Knowing the usage levels of the network allows future requirements to be analyzed.

1. **Loss of Packets:** Occurs when one or more [packets](https://en.wikipedia.org/wiki/Packet_(information_technology)) of data travelling across a [computer network](https://en.wikipedia.org/wiki/Computer_network) fail to reach their destination. Packet loss is either caused by errors in data transmission, typically across [wireless networks](https://en.wikipedia.org/wiki/Wireless_network)**,** Packet loss is measured as a percentage of packets lost with respect to packets sent.

The [Transmission Control Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) (TCP) detects packet loss and performs retransmissions to ensure [reliable messaging](https://en.wikipedia.org/wiki/Reliable_messaging). Packet loss in a TCP connection is also used to [avoid congestion](https://en.wikipedia.org/wiki/TCP_congestion-avoidance_algorithm) and thus produces an intentionally reduced [throughput](https://en.wikipedia.org/wiki/Throughput) for the connection.

1. **Number of students depend on efficiency of processor, also depend on the RAM of device**