# 

KEA\_STUD CHAT MESSENGER

Solution description and baseline cost

[Document information](#h.qwydk7x1nscw)

[Document version](#h.qgvwgi1jjpq)

[Approval List](#h.6r7j8dh65x32)

[Confidentiality Rating](#h.hes98nyepwis)

[General](#h.vsfaotkepuvm)

[Solution summary](#h.obohzlyxwrmy)

[Deliverables summary](#h.l2pd4mvl8bhx)

[Cost summary](#h.uuddz75kzwb4)

[Recommendation and next steps](#h.wcuiy14bok5x)

[Detailed solution description](#h.avkkkztjqpar)

[Technical dictionary](#h.9mwuznd7hrfd)

[Architecture overview](#h.k78al19aaso3)

[Server setup](#h.v8sievqg4cdm)

[Functional requirements](#h.srhnrravenhi)

[Non-functional requirements](#h.cov4a8ojgpfv)

[Capacity recommendations](#h.q4gw0162nchu)

[Impact on other system](#h.ykwmexgv6n1)

[Failover and scalability](#h.ndht1r7b6n3n)

[Technical Implementation](#h.o1bjd21jibx)

[Solution implementation components](#h.7c9cnv780vbj)

[Cost](#h.8x08e4hvqizg)

[Platform cost](#h.28y6hwd99ohr)

[License and support](#h.syha32knd6hd)

[Operational Cost](#h.fm3l13xtid1y)

[Risks](#h.7wc0yv731cjc)

# **Document information**

## Document version

|  |  |  |
| --- | --- | --- |
| Version | Author e-mail | Description |
| 1.0 | Carina Lamb [cari2873@stud.kea.dk](mailto:cari2873@stud.kea.dk)  Dechen Chodon [dech0003@stud.kea.dk](mailto:dech0003@stud.kea.dk)  Lina Alhajar [lina.alhajar@gmail.com](mailto:lina.alhajar@gmail.com)  Muniba Talha [muni0144@stud.kea.dk](mailto:muni0144@stud.kea.dk) | Initial draft |
| 1.1 | Author e-mail | Final Version |
|  | Carina Lamb [cari2873@stud.kea.dk](mailto:cari2873@stud.kea.dk)  Dechen Chodon [dech0003@stud.kea.dk](mailto:dech0003@stud.kea.dk)  Lina Alhajar [lina.alhajar@gmail.com](mailto:lina.alhajar@gmail.com)  Muniba Talha [muni0144@stud.kea.dk](mailto:muni0144@stud.kea.dk) |  |

## Approval List

|  |  |  |
| --- | --- | --- |
| Who | Function | E-mail |
| Nikolaj B. Hemmeshøj | Head of Enterprise Architecture | [nibh@kea.dk](mailto:nibh@kea.dk) |
| Jarl Tuxen | Chief Information Security Officer | [jart@kea.dk](mailto:jart@kea.dk) |

## Confidentiality Rating

|  |  |
| --- | --- |
| Rating |  |
| Company Confidential | X |
| Non Confidential |  |

# **General**

KEA\_STUD Chat messenger will provide the possibility of using a chat within an institute. It will provide the user with the facility to communicate in-group or private, to exchange small/medium files during conversation and save the chat history. In order to run, the chat system will be using a Local Area Network (LAN) connection.



***Fig-1***

## 

## Solution summary

KEA\_STUD LAN chat messenger will provide ease to users in terms of connection, as it will enable users inside the organisation’s firewall in order to connect and communicate to each other using existing resources without having to be connected to the Internet. The communication will be platform independent. Moreover, it will reduce the cost of communication (by minimising mobile/text usage) and the maintenance costs. No centralised server or active internet connection is required for communication.

## Deliverables summary

KEA\_CHAT LAN messenger will deliver following results that can be measured afterwards:

* User Login & Signup options.
* Internet less connection between users.
* Facility to send group and private messages.
* Ability to exchange files during conversation.
* Option to save message history for future referral.
* Ease of communication between KEA students and staff.
* Setup manual

1. How to install server
2. How to setup Database
3. How to get the system going and maintaining

## Cost summary

High level cost elements that the project will carry once implemented or will be running:

* Software development hours - in case of further extension of the software.
* Maintenance cost - in case of system breakdown.

The initial development of KEA\_STUD LAN chat messenger didn’t incur any cost because we utilised already available resources and open source software. High-level cost elements that the project will not carry as it can use existing architecture.

* The Server can run on any existing machines, so it saves the hosting costs.
* Open source software was used for development, so no licence costs.
* Operation cost, as it can be set up using existing resources from KEA and does not require any additional equipment.

# **Recommendation and next steps**

KEA\_STUD LAN chat messenger can be implemented by using one dedicated machine, which will run the Server. Further improvements can be done concerning the implementation of the centralized Database Management System, which could not be done due to shortage of time. The system will help with communication between people during university hours as all the users within the campus can interact with each other without using Internet/Phone calls. The online students can help each other instantly, and administration or teachers can respond to student queries quicker.

Although the private chat option is already available, privacy could be improved, by having an administrator be assigned for authorizing and assigning access privileges to particular users in specific chat groups. Furthermore, the other features like voice/video call could be implemented for enhanced user experience.

As the load increases, more servers and a load balancer could be added to handle the load without compromising user experience.

# **Detailed solution description**

Normally when a student or a staff member at KEA needs to talk to each other within the campus, they call the person in question. This however acquires costs, or by leaving an email then the student/staff member will need to wait for a reply and sometimes it can take up to a day or two to get a reply, which wastes a lot of time and can possibly delay important matters or decisions. In order to prevent this, our KEA\_STUD LAN chat messenger will provide a solution to various communication problems within the institute and save resources, like time.

The strength of the system lies in enabling users to communicate without having internet access. The system allows the users to save their chat histories and allows them to exchange files when required, hence no need to hassle about composing a separate e-mail with an attachment to transfer the file. KEA\_STUD LAN chat messenger supports centralized server mode. The user information and the history will be saved in a MySQL database, which makes the system more optimized and secure as compared to a traditional file system.

## Technical dictionary

## 

Here is a brief explanation of the few terms that have been used in the text:

**Socket:** A socket is an object that represents the low level access to the IP stack. A socket is a simulated medium that allows sending and receiving of data in an application.

**LAN:** A **local area network** (**LAN**) is a computer network that interconnects computers within a limited area, such as a home, school, computer laboratory, or office building, using network media.

**MVC:** Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

**Load Balancer**: A load balancer is a device that acts as a reverse proxy, and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications.

**Server Farm:** A server farm is a collection of computer servers usually maintained by an enterprise to accomplish server needs far beyond the capability of one machine.

## Architecture overview

KEA\_STUD LAN chat messenger is based on the Model View Controller (MVC) architecture in order to make further extension of the application easy and making code reusable. The processing and logic part has been kept separate from the graphical user interface and controllers, i.e. meaning that the application handling for database queries is implemented separately while the user interface presentation and logic processing likewise have been implemented separately. The whole application is further divided into sub applications i.e. client side app and server app. The client application runs on the user’s computer and the server can run on any computer on a network. To send/receive a message the user should be connected to the server. The user can broadcast a message to all the users online (Public Chat) or can send it to any particular user (Private Chat).

The application is developed using Object Oriented Programming in Java Language. To establish the communication link between the systems on the network we need socket connections. A socket enables the application/users to connect to the network and communicate with other applications/users connected to the same network. On a particular machine the socket is composed of an IP address and a port number.

As mentioned above there will be two applications, one for the client and one for the server, so two sockets are made. The client application will execute the client socket, while the server application will run the server socket. To connect to the server socket, the client requires its IP and the port number. The client and server need to share the same port number in order to achieve the connection. Moreover they need to agree upon the protocol used that could be TCP, UDP or RAW. In our case it will use TCP/IP protocol for connection.

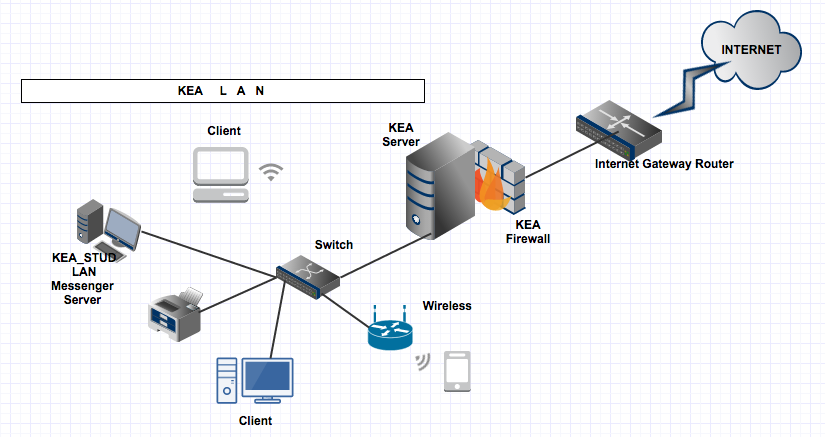


***Fig-2***

## Server setup

Many computers on the network have the capabilities to run the server side of the application while they are executing or running the other services. But it can always be executed on a dedicated workstation to handle the load in a better way. The client knows the IP address and the port of the system on which the server is running. In IP networking context, the server is a program that listens for the socket requests. The client sends the connection request, which the server accepts. Hence, a communication link between client and server is established.

If the number of users increase then additional servers can be added in the architecture, the load balancing can be implemented, which can distribute the connection requests among the servers to handle the requests efficiently.



## Functional requirements

The KEA\_STUD LAN chat messenger will exhibit following functional requirements:

* Register User.
* Login to the System.
* Display list of online contacts.
* Communicate without Internet access.
* View the Online users and groups.
* View chat history.
* User can chat in a group.
* Save chat history.
* Send/Receive small-medium sized files.
* Notification on receiving new message.
* The system must display the notification on receiving reply/new message.

## Non-functional requirements

Here is the list of non-functional requirements that our system should meet:

|  |  |
| --- | --- |
| **Security** | * It is a secure LAN messenger. * The data is stored on MySQL database, which makes it, secure than traditional file system. * Doesn’t support Encryption. |
| **Reliability** | * It cannot accommodate the failure in case the server fails. But the server can be restarted to make it work. * The system should be up and working 95% of the time. |
| **Usability** | * It has user-friendly graphical interface. * It supports English language. |
| **Supportability** | * The complete operation document and installation manual will be provided for the customer support. |
| **Performance** | * The system performs well and the normal response time is less than a second. Even in peak hours, the system should not take more 4-5s to respond. |

## Capacity recommendations

Currently we will require only one machine to run server. When the database is centralised, the database server can be implemented on the same machine. By theoretically assuming that with the passage of time, the number of clients using the system will increase and there comes a point when more than one machine could be required to run the server. In that case, the load-balancing device could be setup to distribute the traffic equally and let the application work optimally during peak hours. Moreover having a distributed server setup and a load balancer will also increase the reliability of the application, as in case of a server break down, the user traffic could be directed to the running server in the server farm.

# **Impact on other system**

This system is not a part of any company or infrastructure so it will not have any impact on any other system or company as of yet. The resources used to implement the system are available within the institute so no external dependency will affect the internal setup or infrastructure.

**Failover and scalability**

***Scalability:*** The system has been developed to keep the option of future upgrading of the client and server open i.e. any component in the system, like the database or the server etc. can be upgraded later to accommodate the increased number of users.

***Failover:*** To prevent failover and keeping system up and running 95% of the year multiple servers will be working, so if one of the server will crash the other servers will be there to provide the service.

**Technical implementation plan**

The system is built in Eclipse IDE for Java developers. Java development kit (jdk) is required to compile and run the program. The object-oriented programming using java language is used to write the source code for the system. The database is implemented in MySQL.

## Solution implementation components

### Preparation

1. Understanding why system should be built.
2. Analysis of functional and non-functional requirements.
3. Define the scope of the project.
4. Create installation manuals.
5. Create feasibility plan.

### Development of software

1. Software design implementation.
2. Database designing.
3. Building algorithms to implement the design.
4. Writing the source code of the system.
5. System testing and removing bugs.

### 

### Hardware setup

1. Install Servers
2. Install Eclipse IDE
3. Install jdk 1.5 or later
4. Install databases

**Cost**

Initially the system will not cost anything as it has been built by using open source software. The Eclipse IDE is used for development and MySQL is used to implement the database, and we recommend using the future extension of this software, as these days many open source software are readily available with equally powerful features for developers and it will save lots of capital.

## Platform cost

For server implementation, we can use one powerful stand-alone machine or various economical machines, which can set up in a server farm and resources can be accessed using a load balancer, which distributes the user load and will help save money.

## License and support

All products used for the development of this application have either GPL or EPL open source licenses. Support will be available in the form of the user manual and operation documents. Moreover, the system is not very complex and due to its GUI, it is easy to use. The network administrator already working in the organization can be equipped to provide support in case if unexpected issues arise.

## Operational Cost

Initial operational cost will be zero because we will be using existing resources but, as the number of active users will increase, then multiple servers will be required to set up the database and the server. Cost can depend on the choice of machine and the number of active users.

Moreover, the hardware purchased might not have an operating system, which can be purchased from prices starting at $1000 and up to $4000 for an Enterprise edition Windows Server. Similarly, a MySQL database server will cost $900 or more.

# **Risks**

When it comes to small - medium level development using a limited period of time and resources, there is always a possibility of the risks coming your way when the application becomes operational. Keeping this fact in mind, we have developed a proactive risk strategy to identify the risks and develop preventive measures against it.

The possible risks for the project have been identified, and depending on the frequency of their occurrence they have been scaled between 1-10 and at last a mitigation action and solution has been developed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item # | Area | Description | Rank | Mitigation | Solution |
| #1 | Technical | Server Breakdown | 4 | Monitor Server Load | Multiple Servers |
| #2 | Software | Run time error | 5-6 | JRE missing | Install JRE |
| #3 | Technical | Power Interruption | 2 | Sudden power failure | UPS installation |
| #4 | System | System fails under stress | 3-4 | Scaling up the system | Load balancer to distribute the users in server farm |