

A MINOR PROJECT REPORT

On

Work Allocator

Submitted in partial fulfillment of the requirement for the award of the
degree of

B.TECH

in

COMPUTER SCIENCE AND ENGINEERING

Submitted By

RAHUL SUBHAGAN : RA1611003040210

SAEM AHMER : RA1611003040251

P.BALA : RA1611003040198



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPT. OF COMPUTER SCIENCE & ENGINEERING

SRM Institute of Science & Technology
Vadapalani Campus, Chennai

OCTOBER 2018



BONAFIDE CERTIFICATE

Certified that this project report "**WORK ALLOCATOR**" is the bonafide work of "**RAHUL SUBHAGAN [Reg.No. RA1611003040210], SAEM AHMER[Reg.No.RA16110030400251],P.BALA[Reg.No.RA1611003040198]**" who carried out the project work under my supervision.

SIGNATURE OF THE GUIDE

Mr Sridhar S, B.Tech, M.E.

Assistant Professor

Department of Computer Science and
Engineering

SRM Institute of Science & Technology
Vadapalani Campus

SIGNATURE OF THE HOD

Dr.S.Prasanna Devi, B.E.,M.E., Ph.D.,
PGDHRM.,PDF(IISc)

Professor

Department of Computer Science and
Engineering

SRM Institute of Science & Technology
Vadapalani Campus

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, Mr. Sridhar S. for her valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Dr .S.Prasanna Devi, for encouraging and allowing us to present the project on the topic “WORK ALLOCATOR “ at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

We take this opportunity to thank all our faculty members, Dean, Dr.K.Duraivelu, and Management who have directly or indirectly helped our project. Last but not the least we express our thanks to our friends for their cooperation and support.

WORK ALLOCATOR

ABSTRACT

Employee Work Management using Java and SQL Server

In this world of growing technologies everything has been computerized. With large number of work opportunities the Human workforce has increased. Thus there is a need of a system which can handle the data of such a large number of Employees in an organization. This project simplifies the task of maintaining records and assigning projects with help of its user friendly nature.

The system makes this process of scheduling much easier and computerized. By this system the manager or top level designated employee can assign projects and works for any employee working under him, a deadline will be given for the particular work assigned to the employee and if the work is submitted before the deadline a progress report will be evaluated by the supervisor of the respective field.

Thus the top level management can easily fix the process of assigning tasks and even can change the appointment which is reflected immediately to the related employee avoiding direct contact of the employee resulting in saving lot of time and work overhead.

Contents:

Chapter -1- INTRODUCTION

1.1 Overview.....	08
1.2 Existing system	08
1.3 Motivation.....	09

Chapter -2- PROJECT WORK

2.1 Scope of the project.....	10
2.1.2 Knowing around.....	10
2.2 Java Swing Applications	
2.2.1 Eclipse Jee	10
2.2.2 Java Swing.....	11
2.2.3 JFrame	12
2.2.4 Swing Components	12
2.3 Structured Query Language	
2.3.1 Overview	13
2.3.2 Sql For Database	15
2.3.3 phpMyAdmin.....	15
2.4 XAMPP	
2.4.1 Overview	17
2.4.2 Local Hosting	18
2.4.3 Ports.....	20
2.5 Java Database Connectivity	
2.5.1 Overview	21
2.5.2 MySQL Driver Packages.....	23

Chapter -3- SOURCE CODE AND SCREENSHOTS

3.1 Source code	24
3.2 Screenshots	41

Chapter -4- CONCLUSION AND FUTURE SCOPE

4.1 Conclusion	44
4.2 Future Scope	44

Chapter -5- REFERENCES45

LIST OF FIGURES

2.1	JFrame	12
2.2	Swing Components	12
2.3	phpMyAdmin database.....	15
3.1	XAMPP	17
4.1	Manager login.....	42
4.2	Manager panel.....	42
4.3	Employee login	25

CHAPTER 1

INTRODUCTION

1.1 Overview

The average employee nowadays leads a very hectic life. They have to deal with many submissions and deadlines. In general we can say that from an employee's perspective that the modern companies are full of demands, frustrations, hassles and deadlines. Even from a manager's perspective, every work requires a lot of planning and a lot of brainstorming. So, when any work is being assigned to an employee, a manager generally has to end up continuously updating himself about the regular work done by the employee. By our application the Work Allocator we intend to make this work a lot easier for the manager as well as the employee, with continuous updates of the progress done by the employee for the specific work assigned to him.

With the work allocator, both the employee and manager can easily balance their time and it helps with a lot of unnecessary physical confrontation. Due to this application the amount of miscommunication between the employee and the manager is avoided, the manager can simply allot the work to his employees and can regularly check progress made, following the deadline required to complete it.

With this project, we are aiming to reduce this time that it takes for the employee and the manager to interact to a bare minimum. Keeping a manual check if the employee is as dedicated for the work or not.

1.2 Existing system

The current system in place for assigning work is completely manual. The manager has to first assign a task which is usually done on paper or in verbal meetings. Even after this, the process requires the employee to inform about the progress made by him on a regular

basis in meeting with takes up a lot of time , time which can be easily avoided by the work allocator .We tend to reduces all these formalities for work being assigned and makes this work allocating and reviewing less time consuming.

Finding time for discussing each and every progress made by the employee , and with many employee's working under a single manager . Checking up on every employee each and every work assigned to him/her in a huge company requires a lot of time and sometimes even results in the employee after completing the whole process ends up not being as productive as he could have been.

Also, on the managers side maintaining each and every progress of the employee it quite a difficult task. With a lot on the manager's plate already physically checking up on the employee takes up a lot of toll for the manger and even the employee may find it quite tiresome , repetitive , physically wearing as well as psychologically stressful and it only seems fair that the workload for them both would be reduced from this.

1.3 Motivation

Everyday many tasks are assigned by a company to their employees , this requires a lot of documentation procedure and lots of time which can be redirected to completing the task at hand with greater efficiency and produce more better results.

CHAPTER 2

PROJECT WORK

2.1 Scope of the project

Although the current system tends to be satisfying the requirements of manger and they still get their work sanctioned, it involves a lot of manual work. Our project concentrates on getting the work assigned with much less efforts required to get it and increasing the time the employee has to work on the task.

The main objective of this project is to optimize the efforts put in the process of obtaining the work by just reducing the work to just filling out a form in our application. This work assigning becomes a lot simpler as well as the progress which reduces the total time of the process to just mere seconds. This ease in the total process makes the process seem incomplete but actually the process is completely getting optimized. This project is also environment friendly as it reduces the usage of paper on a quite large scale as the complete process is shifted to your fingertips in the form of an app.

2.1.2 Knowing around

We will start our Java Swing application development on any of the following operating systems:

Microsoft Window XP or later version.

Mac OS X 10.5.8 or later version with Intel chip Linux including GNU Library with Intel chip.

2.2 Java swing Applications

2.2.1 Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages

via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, D, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the OSGi implementation (Equinox) in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundles.

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License. It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

2.2.2 Java Swing

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element

Swing is a highly modular-based architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations using Java's inheritance mechanism.

Swing is a component-based framework, whose components are all ultimately derived from the `javax.swing.JComponent` class. Swing objects asynchronously fire events, have bound properties, and respond to a documented set of methods specific to the component. Swing components are Java Beans components, compliant with the Java Beans Component Architecture specifications.

2.2.3 JFrame

A `JFrame` is a component and top-level container of the JFC/Swing framework. A `JFrame` is a `Frame` element in Java Swing but is slightly incompatible with `Frame`. It is normally the outermost component, has a `Frame` and a title, and is usually filled with menu and `JPanel(s)` which contain(s) further components.

`JFrame` is heavyweight, since it's based on creating a "heavyweight" AWT window. Lightweight components can replace internal widgets with java-based implementation that doesn't require the use of JNI (Java Native Interface), but windows are the special case. `JFrame` does let you do custom rendering, via its heavy window. Also, if you're using other lightweight stuff, all of them will be added to the `ContentPane`. Using `JFrame` makes the rendering more efficient overall than mixing light and heavy components. `JFrame` itself is a top-level container and contains a `JRootPane` as its only child. The content pane provided by the root pane should, as a rule, contain all the non-menu components displayed by the `JFrame`. This is different from the AWT `Frame` case.

2.2.4 Swing components

A component is an independent visual control. Swing Framework contains a large set of components which provide rich functionalities and allow high level of customization. They all are derived from `JComponent` class. All these components are lightweight components. This class provides some common functionality like pluggable look and feel, support for accessibility, drag and drop, layout, etc.

A container holds a group of components. It provides a space where a component can be managed and displayed.

JButton class provides functionality of a button. `JButton` class has three constructors, It allows a button to be created using icon, a string or both. `JButton` supports **ActionEvent**. When a button is pressed an **ActionEvent** is generated.

JTextField is used for taking input of single line of text. It is most widely used text component. It has three constructors, *cols* represent the number of columns in text field.

JCheckBox class is used to create checkboxes in frame. Following is constructor for JCheckBox,

JRadioButton is a group of related button in which only one can be selected. JRadioButton class is used to create a radio button in Frames.

JComboBox is a combination of text fields and drop-down list. **JComboBox** component is used to create a combo box in Swing.

JLabel

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

JPasswordField

The object of JPasswordField class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

JScrollBar

The object of JScrollbar class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits JComponent class.

2.3 Structured query language(SQL)

2.3.1 Overview

Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data where there are relations between different entities/variables of the data. SQL offers two main advantages over older read/write APIs like ISAM or VSAM: first, it introduced the concept of accessing many records with one single command; and second, it eliminates the need to specify how to reach a record, e.g. with or without an index.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements, which may be informally classed as sublanguages, commonly: a data query language (DQL),[a] a data definition language (DDL),[b] a data control language (DCL), and a data manipulation language (DML).The scope of SQL includes data query, data manipulation (insert,

update and delete), data definition (schema creation and modification), and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce after learning about the relational model from Ted Codd in the early 1970s. This version, initially called SEQUEL (Structured English Query Language), was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970s.

Chamberlin and Boyce's first attempt of a relational database language was Square, but it was difficult to use due to subscript notation. After moving to the San Jose Research Laboratory in 1973, they began work on SEQUEL. The acronym SEQUEL was later changed to SQL because "SEQUEL" was a trademark of the UK-based Hawker Siddeley aircraft company.

In the late 1970s, Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Codd, Chamberlin, and Boyce, and developed their own SQL-based RDBMS with aspirations of selling it to the U.S. Navy, Central Intelligence Agency, and other U.S. government agencies. In June 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers. By 1986, ANSI and ISO standard groups officially adopted the standard "Database Language SQL" language definition. New versions of the standard were published in 1989, 1992, 1996, 1999, 2003, 2006, 2008, 2011, and most recently, 2016. After testing SQL at customer test sites to determine the usefulness and practicality of the system, IBM began developing commercial products based on their System R prototype including System/38, SQL/DS, and DB2, which were commercially available in 1979, 1981, and 1983, respectively.

SQL deviates in several ways from its theoretical foundation, the relational model and its tuple calculus. In that model, a table is a set of tuples, while in SQL, tables and query results are lists of rows: the same row may occur multiple times, and the order of rows can be employed in queries (e.g. in the LIMIT clause).

Critics argue that SQL should be replaced with a language that strictly returns to the original foundation: for example, see The Third Manifesto. However, no known proof exists that such uniqueness cannot be added to SQL itself, or at least a variation of SQL. In other words, it's quite possible that SQL can be "fixed" or at least improved in this regard such that the industry may not have to switch to a completely different query language to obtain uniqueness. Debate on this remains open.

2.3.2 SQL for Database

An SQL database is a relational database that makes use of the Structured Query Language (SQL) to query or communicate with the database.

A database is a collection of data organized in tables. A flat database is where all the information about any one event or transaction is stored in a single row in one large single table. On the other hand, in an SQL database, data is separated into tables. This helps to breakup and organize the data. The data from the different tables can be combined and extracted using a language called Structured Query Language or SQL.

An SQL database is also sometimes called a Relational database. In an SQL database, data is stored in multiple tables instead of in a single row. The different tables are related or connected using a common field called the key that is common to one or more tables. Using SQL, the data from multiple tables can be combined to get the information that we need.

If the library data in the above example is stored in a relational database, we could break up the long row of data into multiple tables. In a relational database, one table can contain the student contact information, and the other table can contain information about the books. Both tables can then be linked by a common field or key so that we can connect the data from the two tables to find a list of all the books borrowed by Joe.

2.3.3 phpMyAdmin

Features provided by the program include:

- ✓ Web interface

- ✓ MySQL and MariaDB database management
- ✓ Import data from CSV and SQL
- ✓ Export data to various formats: CSV, SQL, XML, PDF (via the TCPDF library), ISO/IEC 26300
- OpenDocument Text and Spreadsheet, Word, Excel, LaTeX and others
- ✓ Administering multiple servers
- ✓ Creating PDF graphics of the database layout
- ✓ Creating complex queries using query-by-example (QBE)
- ✓ Searching globally in a database or a subset of it
- ✓ Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link
- ✓ Live charts to monitor MySQL server activity like connections, processes, CPU/memory usage, etc.
- ✓ Working with different operating systems.
- ✓ Make complex SQL queries easier

1. Features provided by the program include:

2. Web interface
3. MySQL and MariaDB database management
4. Import data from CSV and SQL
5. Export data to various formats: CSV, SQL, XML, PDF (via the TCPDF library), ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, Excel, LaTeX and others
6. Administering multiple servers
7. Creating PDF graphics of the database layout
8. Creating complex queries using query-by-example (QBE)
9. Searching globally in a database or a subset of it
10. Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link
11. Live charts to monitor MySQL server activity like connections, processes, CPU/memory usage, etc.

12. Working with different operating systems.
13. Make complex SQL queries easier
14. Live charts to monitor MySQL server activity like connections, processes, CPU/memory usage, etc.
15. Working with different operating systems.
16. Make complex SQL queries easier

2.4 XAMPP

2.4.1 Overview

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

XAMPP's ease of deployment means a WAMP or LAMP stack can be installed quickly and simply on an operating system by a developer, with the advantage a number of common add-in applications such as Wordpress and Joomla! can also be installed with similar ease using Bitnami .

The term XAMPP is an apparent acronym. However, there is no official acronym expansion specified on the Apache Friends website. Their homepage header reads "XAMPP Apache + MariaDB + PHP + Perl", indicating that this abbreviation is a recursive acronym.

The term can be unofficially broken down as follows:

Letter	Meaning
--------	---------

X	as an ideographic letter referring to cross-platform
A	Apache or its expanded form, Apache HTTP Server
M	MariaDB (formerly: MySQL)
P	PHP
P	PERL

MySQL was replaced with MariaDB on 2015-10-19 and beginning with XAMPP versions 5.5.30 and

5.6.14, effectively altering the meaning of the acronym.

While both letters P are de facto interchangeable, convention used at the Apache Friends website indicates that the first letter P is short for PHP and the latter letter P is short for Perl.

The most obvious characteristic of XAMPP is the ease at which a WAMP webserver stack could be deployed and got running. Later some common packaged applications that could be easily installed were provided by Bitnami.

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. To make this as easy as possible, many important security features are disabled by default. XAMPP has the ability to serve web pages on the World Wide Web. A special tool is provided to password-protect the most important parts of the package.

XAMPP also provides support for creating and manipulating databases in MariaDB and SQLite among others.

Once XAMPP is installed, it is possible to treat a localhost like a remote host by connecting using an FTP client. Using a program like FileZilla has many advantages when installing a content management system (CMS) like Joomla or WordPress. It is also possible to connect to localhost via FTP with an HTML editor.

2.4.2 Local Hosting

The name may also be resolved by Domain Name System (DNS) servers, but queries for this name should be resolved locally, and should not be forwarded to remote name servers.

In addition to the mapping of localhost to the loopback addresses (127.0.0.1 and ::1), localhost may also be mapped to other IPv4 (loopback) addresses and it is also possible to assign other, or additional, names to any loopback address. The mapping of localhost to addresses other than the designated loopback address range in the hosts file or in DNS is not guaranteed to have the desired effect, as applications may map the name internally.

In the Domain Name System, the name localhost is reserved as a top-level domain name, originally set aside to avoid confusion with the hostname used for loopback purposes. IETF standards prohibit domain name registrars from assigning the name localhost.

The name may also be resolved by Domain Name System (DNS) servers, but queries for this name

should be resolved locally, and should not be forwarded to remote name servers.

In addition to the mapping of localhost to the loopback addresses (127.0.0.1 and ::1), localhost may also be mapped to other IPv4 (loopback) addresses and it is also possible to assign other, or additional, names to any loopback address. The mapping of localhost to addresses other than the designated loopback address range in the hosts file or in DNS is not guaranteed to have the desired effect, as applications may map the name internally.

In the Domain Name System, the name localhost is reserved as a top-level domain name, originally set aside to avoid confusion with the hostname used for loopback purposes. IETF standards prohibit domain name registrars from assigning the name localhost.

The name may also be resolved by Domain Name System (DNS) servers, but queries for this name should be resolved locally, and should not be forwarded to remote name servers.

In addition to the mapping of localhost to the loopback addresses (127.0.0.1 and ::1), localhost may also be mapped to other IPv4 (loopback) addresses and it is also possible to assign other, or additional, names to any loopback address. The mapping of localhost to addresses other than the designated loopback address range in the hosts file or in DNS is not guaranteed to have the desired effect, as applications may map the name internally.

In the Domain Name System, the name localhost is reserved as a top-level domain name, originally set aside to avoid confusion with the hostname used for loopback purposes. IETF standards prohibit domain name registrars from assigning the name localhost. The name may also be resolved by Domain Name System (DNS) servers, but queries for this name should be resolved locally, and should not be forwarded to remote name servers.

In addition to the mapping of localhost to the loopback addresses (127.0.0.1 and ::1), localhost may also be mapped to other IPv4 (loopback) addresses and it is also possible to assign other, or additional, names to any loopback address. The mapping of localhost to addresses other than the designated loopback address range in the hosts file or in DNS is not guaranteed to have the desired effect, as applications may map the name internally.

In the Domain Name System, the name localhost is reserved as a top-level domain name, originally set aside to avoid confusion with the hostname used for loopback purposes. IETF standards prohibit domain name registrars from assigning the name localhost. The name may also be resolved by Domain Name System (DNS) servers, but queries for this name should be resolved locally, and should not be forwarded to remote name servers.

In addition to the mapping of localhost to the loopback addresses (127.0.0.1 and ::1), localhost may also be mapped to other IPv4 (loopback) addresses and it is also possible to assign other, or additional, names to any loopback address. The mapping of localhost to addresses other than the designated loopback address range in the hosts file or in DNS is not guaranteed to have the desired effect, as applications may map the name internally.

2.4.3 Ports

In computer networking, a port is an endpoint of communication. Physical as well as wireless connections are terminated at ports of hardware devices. At the software level, within an operating system, a port is a logical construct that identifies a specific process or a type of network service.

The software port is always associated with an IP address of a host and the protocol type of the communication. It completes the destination or origination network address of a message. Ports are identified for each protocol and address combination by 16-bit unsigned numbers, commonly known as the port number.

Ports provide a multiplexing service for multiple services or multiple communication sessions at one network address. Specific port numbers are commonly reserved to identify specific services. The lowest numbered 1024 port numbers are called the well-known port numbers, and identify the historically most commonly used services. In the client–server model of application architecture, a multiplexing service is established, so that multiple simultaneous communication sessions may be initiated for the same service. The most commonly used protocols that use ports are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

An example for the use of ports is the Internet mail system. A server used for sending and receiving email generally needs two services. The first service is used to transport email to and from other servers. This is accomplished with the Simple Mail Transfer Protocol (SMTP). The SMTP service application usually listens on TCP port 25 for incoming requests. The second service is usually either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP) which is used by e-mail client applications on users' personal computers to fetch email messages from the server. The POP service listens on TCP port number 110. Both services may be running on the same host computer, in which case the port number distinguishes the service that was requested by a remote computer, be it a user's computer or another mail server.

While the listening port number of a server is well defined (IANA calls these the well-known ports), the client's port number is often chosen from the dynamic port range (see below). In some

applications, the clients and the server each use specific port numbers assigned by the IANA. A good example of this is DHCP in which the client always uses UDP port 68 and the server always uses UDP port 67.

2.5 Java Database Connectivity

2.5.1 Overview

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

Statement – the statement is sent to the database server each and every time.

PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

CallableStatement – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the javax.sql.

JDBC connections are often managed via a connection pool rather than obtained directly from the driver.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC

connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

Statement – the statement is sent to the database server each and every time.

PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

CallableStatement – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the javax.sql.

JDBC connections are often managed via a connection pool rather than obtained directly from the driver.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

Statement – the statement is sent to the database server each and every time.

PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

CallableStatement – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the javax.sql.

JDBC connections are often managed via a connection pool rather than obtained directly from the driver.

2.5.2MySql Driver packages

The interface for accessing relational databases from Java is Java Database Connectivity (JDBC). Via JDBC you create a connection to the database, issue database queries and update as well as receive the results. JDBC provides an interface which allows you to perform SQL operations independently of the instance of the used database. To use JDBC, you require the database specific implementation of the JDBC driver.

The following description will assume that you have successfully installed MySQL and know how to access MySQL via the command line. To connect to MySQL from Java, you have to use the JDBC driver from MySQL. The MySQL JDBC driver is called *MySQL Connector/J*.

CHAPTER 3

SOURCE CODE AND SCREENSHOTS

3.1 SOURCE CODE

Mlogin.java

```
package mysql;

import java.awt.BorderLayout;

public class Mlogin extends JFrame {

    /**
     * Launch the application.
     */

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Mlogin frame = new Mlogin();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```



```

    }

    });

}

/**
 * Create the frame.
 */

public Mlogin() {

    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JButton btnNewButton = new JButton("MANAGER");

    btnNewButton.setBackground(SystemColor.activeCaptionBo
rder);

}

}

```

Login.java

```

package mysql;

import java.sql.*;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.EventQueue;

        pass = new JPasswordField();

        pass.setBounds(18, 219, 306, 35);

        contentPane.add(pass);

        JButton btnLogin = new JButton("Login");
        btnLogin.setBackground(SystemColor.activeCaption

```

```
Border);
```

```
}
```

```
}
```

Manlog.java

```
package mysql;
```

```
import java.awt.BorderLayout;
```

```
import java.awt.EventQueue;
```

```
        JButton btnLogin = new JButton("Login");
```

```
        public void
```

```
actionPerformed(ActionEvent arg0) {
```

```
            try {
```

```
                Class.forName("com.mysql.jdbc.Driver");
```

```
                Connection
```

```
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/data1","root","");
```

```
                Statement
```

```
stmt=con.createStatement();
```

```
                String sql="Select *
```

```
from table2 where username='"+user.getText()+"' and  
password='"+pass.getText().toString()+"';
```

```
                ResultSet
```

```
rs=stmt.executeQuery(sql);
```

```
                if(rs.next())
```

```
                {
```

```

        JOptionPane.showMessageDialog(null, "Login
        Succesful");

        manager s=new
        manager();

        s.setVisible(true);

    }

    else

        JOptionPane.showMessageDialog(null,"Incorrect
        Username or Password...");

        con.close();

    } catch(Exception e)

    {System.out.print(e);}

    }

    });

    contentPane.add(btnLogin);

}

}

```

About.java

```

package mysql;

    JTextArea textArea = new JTextArea();

    textArea.setBackground(new Color(255, 222,
    173));

    String text = "THIS IS BASICALLY AN
    EMPLOYEE WORK MANAGEMENT SYSTEM.\n"

    + "THIS IS IDEALISED TO

```

CREATE A SMOOTH WORKING ENVIRONMENT
IN A BUSY SCHEDULED COMPANY.\n"

+ "THROUGH THIS TOOL A
TOP DESIGNATED MANAGER COULD MANAGE
ALL THE EMPLOYEE ON WORK ALLOCATION,\n"

+ "SUPERVISION AND
MOST IMPORTANTLY TIME MANAGEMENT .\n"

+ "SO BASICALLY A
MANAGER COULD KEEP AN EYE ON THE
STATUS OF ALL ONGOING PROJECTS UNDER
HIM.";

```
        textArea.setText(text);  
  
        scrollPane.setViewportView(textArea);  
  
    }  
}
```

sec.java

```
package mysql;  
  
import java.awt.BorderLayout;  
  
import java.awt.EventQueue;  
  
public static void main(String[] args) {  
  
    EventQueue.invokeLater(new Runnable() {  
  
        public void run() {  
  
            try {  
  
                sec frame = new sec();  
  
                frame.setVisible(true);  
  
            } catch (Exception e) {
```

```

                                e.printStackTrace();
                                }
                                }
                                });
                                }

public sec(String st) {

    int a;

    String o;

    if(st.equals("Saem"))

        a=1;

    else if(st.equals("Bala"))

        a=2;

    else

        a=3;

    if(a==1)

        o="1";

    else if(a==2)

        o="2";

    else

        o="3";

    btnSubmit.addActionListener(new ActionListener() {

        public void

        actionPerformed(ActionEvent arg0) {

            try {

                Class.forName("com.mysql.jdbc.Driver");

```

Connection

```
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/data1","root","");
```

Statement

```
stmt=con.createStatement();
```

String

```
k=TextArea_1.getText();
```

```
    stmt.executeUpdate("insert into scode  
(empcode,temp) values('"+k+"','"+o+"')");
```

```
    con.close();
```

```
    }catch(Exception e) {System.out.print(e);} 
```

```
    }
```

```
});
```

```
btnSubmit.setBounds(668, 477, 97, 25);
```

```
contentPane.add(btnSubmit);
```

```
        JButton btnClearAll = new JButton("Clear  
Submissions");
```

```
        btnLoad = new JButton("Load");
```

```
        btnLoad.setBackground(SystemColor.activeCaptionB  
order);
```

```
        btnLoad.addActionListener(new  
ActionListener() {
```

```
            public void
```

```
actionPerformed(ActionEvent arg0) {
```

```
                try {
```

```

        Class.forName("com.mysql.jdbc.Driver");

        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/data1","root","");

        Statement

stmt=con.createStatement();

        String sql;

        if(a==1)

            sql="Select empcode from
score where temp=1";

        else if(a==2)

            sql="Select empcode
from score where temp=2";

        else

            sql="Select empcode
from score where temp=3";

        public sec() {

            // TODO Auto-generated constructor stub

        }

}

```

manager.java

```

package mysql;

import java.awt.BorderLayout;

import java.awt.EventQueue;

        public void
actionPerformed(ActionEvent arg0) {

```

```

        String st="Bala";

        View s=new View(st);

        s.setVisible(true);

    }

});

btnView.setBounds(313, 122, 97, 25);

contentPane.add(btnView);


JLabel lblSaemAhmer = new JLabel("Saem
Ahmer");

lblSaemAhmer.setBounds(110, 190, 87, 16);

contentPane.add(lblSaemAhmer);


JButton btnView_1 = new JButton("VIEW");


btnView_1.setBackground(SystemColor.activeCaptionBorder);

btnView_1.addActionListener(new
ActionListener() {

    public void
actionPerformed(ActionEvent e) {

        String st="Saem";

        View s=new View(st);

        s.setVisible(true);

    }

});

```



```

        btnView_1.setBounds(313, 186, 97, 25);

        contentPane.add(btnView_1);

        JLabel lblNewLabel = new JLabel("Rahul
Subhagan");

        btnView_2.addActionListener(new
        ActionListener() {

            public void
            actionPerformed(ActionEvent arg0) {

                String st="Rahul";

                View s=new View(st);

                s.setVisible(true);

            }

        });

        btnView_2.setBounds(313, 247, 97, 25);

        contentPane.add(btnView_2);

    }

}

```

View.java

```

package mysql;

import java.awt.BorderLayout;

public class View extends JFrame {

    private JPanel contentPane;

    public static void main(String[] args) {

```

```

       .EventQueue.invokeLater(new Runnable() {

            public void run() {

                try {

                    View frame = new
View();

                    frame.setVisible(true);

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        });

    }

    public View(String p) {

        int a;

        if(p.equals("Saem"))

            a=1;

        else if(p.equals("Bala"))

            a=2;

        else

            a=3;

        btnNewButton.addActionListener(new
ActionListener() {

            public void

```

```

actionPerformed(ActionEvent arg0) {

    try {

        Class.forName("com.mysql.jdbc.Driver");

        Connection
con=DriverManager.getConnection("jdbc:mysql://localh
ost:3306/data1","root","");

        Statement

stmt=con.createStatement();

        String sql2;

        if(a==1)

            sql2="Select problem from
massign where temp=1";

        else if(a==2)

            sql2="Select problem from
massign where temp=2";

        else

            sql2="Select problem from
massign where temp=3";

        ResultSet

rs2=stmt.executeQuery(sql2);

        while(rs2.next())

        {

            String

p2=rs2.getString("problem");

            textArea.setText(p2);
        }
    }
}

```

```

        con.close();

    } catch (Exception e)
    { System.out.print(e);}

    }

});

}

public View() {

    // TODO Auto-generated constructor stub

}

}

```

Assign.java

```

package mysql;

import java.awt.BorderLayout;

import java.awt.EventQueue;

public class Assign extends JFrame {

    private JPanel contentPane;

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            public void run() {

                try {

                    Assign frame = new

Assign();

                    frame.setVisible(true);

```

```

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

});

}

```

```

public Assign(String p) {

```

```

    int a;

```

```

    String o;

```

```

    if(p.equals("Saem"))

```

```

        a=1;

```

```

    else if(p.equals("Bala"))

```

```

        a=2;

```

```

    else

```

```

        a=3;

```

```

    if(a==1)

```

```

        o="1";

```

```

    else if(a==2)

```

```

        o="2";

```

```

    else

```

```

        o="3";

```

```

setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

setBounds(100, 100, 778, 522);

contentPane = new JPanel();

contentPane.setBackground(new Color(205,
133, 63));

contentPane.setBorder(new EmptyBorder(5, 5,
5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);


JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(23, 90, 713, 274);

contentPane.add(scrollPane);


JTextArea textArea = new JTextArea();

textArea.setBounds(700, 159, 736, 298);

scrollPane.setViewportView(textArea);


JButton btnSubmit = new JButton("Submit");

btnSubmit.addActionListener(new
ActionListener() {

    public void
actionPerformed(ActionEvent arg0) {

        try {

```

```

        Class.forName("com.mysql.jdbc.Driver");

        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/data1","root","");

        Statement

stmt=con.createStatement();

        String

k=TextArea.getText();

        stmt.executeUpdate("insert into massign
(problem,temp) values('"+k+"','"+o+"')");

        con.close();

    } catch(Exception e)
{ System.out.print(e);}

    }

});

actionPerformed(ActionEvent arg0) {

    try {

        Class.forName("com.mysql.jdbc.Driver");

        Connection

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/data1","root","");

        Statement

stmt=con.createStatement();

```

```

        String sql;

        if(a==1)

            sql="Select problem from
massign where temp=1 ";

            else if(a==2)

                sql="Select problem from
massign where temp=2";

            else

                sql="Select problem from
massign where temp=3";

        ResultSet

rs=stmt.executeQuery(sql);

        while(rs.next())

        {

            String

p=rs.getString("problem");

            textArea.setText(p);

        }

        con.close();

    }catch(Exception e)

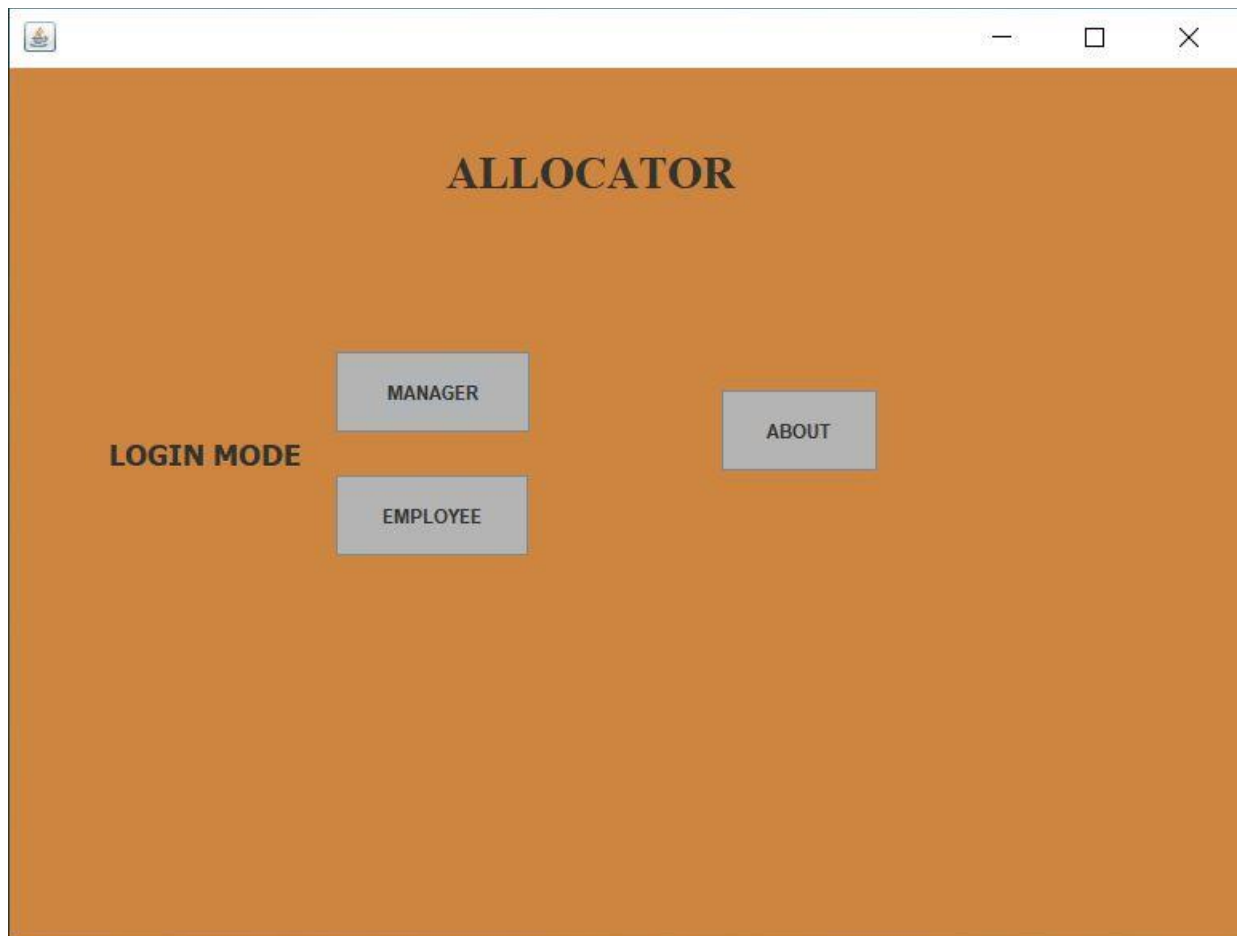
{ System.out.print(e);}

    }

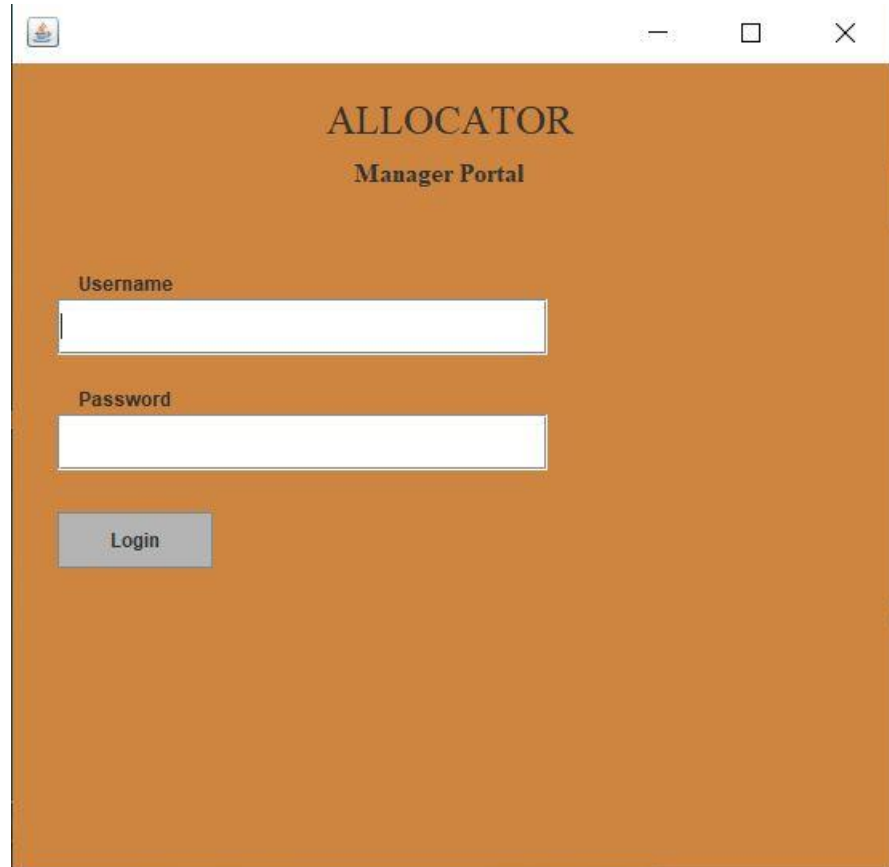
});

```


3.2 OUTPUT SCREENSHOTS



LOGIN MODE



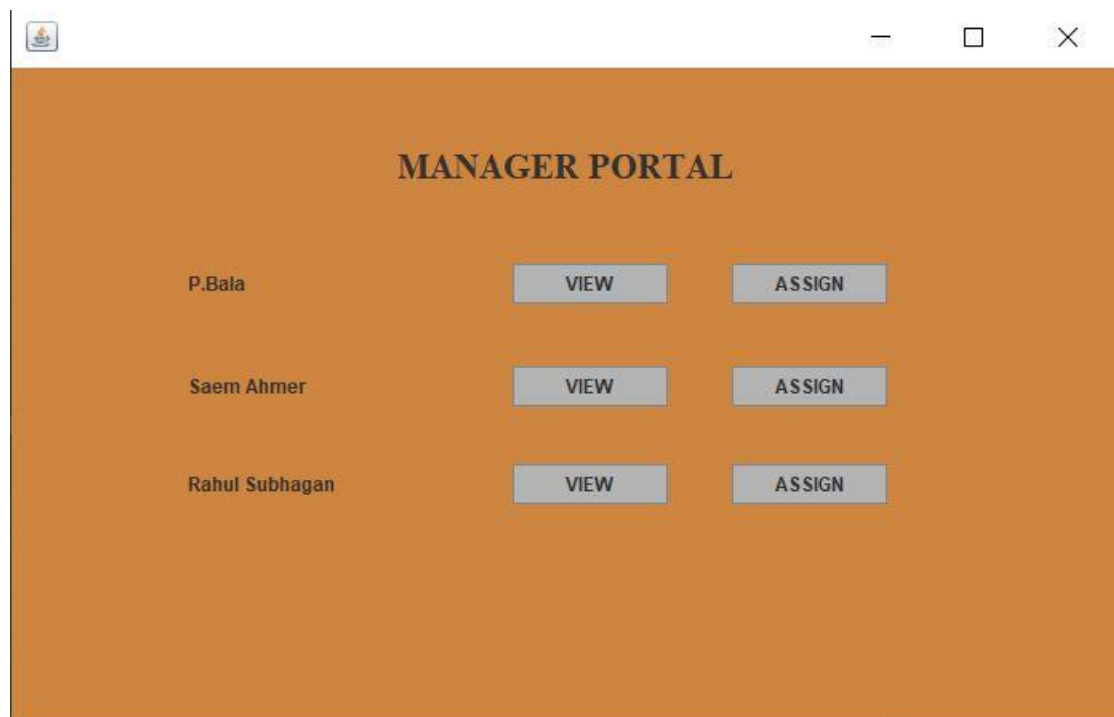
A screenshot of a web application window titled "ALLOCATOR Manager Portal". The window has a standard OS-style title bar with a small icon on the left and minimize, maximize, and close buttons on the right. The background is a solid orange color. In the center, the text "ALLOCATOR" is displayed in a large, bold, serif font, with "Manager Portal" in a smaller, bold, serif font directly below it. Below the title, there are two white input fields with orange borders. The first field is labeled "Username" and the second is labeled "Password". Below these fields is a grey button with the text "Login" in a bold, sans-serif font.

ALLOCATOR
Manager Portal

Username

Password

Login



A screenshot of a web application window titled "MANAGER PORTAL". The window has a standard OS-style title bar with a small icon on the left and minimize, maximize, and close buttons on the right. The background is a solid orange color. In the center, the text "MANAGER PORTAL" is displayed in a large, bold, serif font. Below the title, there is a table with three rows of user information. Each row contains a user name, a "VIEW" button, and an "ASSIGN" button. The buttons are grey with bold, sans-serif text.

MANAGER PORTAL

P.Bala	VIEW	ASSIGN
Saem Ahmer	VIEW	ASSIGN
Rahul Subhagan	VIEW	ASSIGN


— □ ×

ALLOCATING PORTAL

Clear submissions

Load

Submit

— □ ×

PROJECT AREA(Saem)

Load

Clear Submissions

Load

Submit

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

4.1 Conclusion

In this project, we have solved the struggle of work allocation . This has been done by replacing the pen paper work by latest available technology. This proves the value of time by channeling them to something valuable. The application will make the Approval of work faster and better thereby need not worrying about the time constraint.

4.2 Future scope

We look forward to introduce this for institutions and colleges where assigning work is also a hectic task , while applying the same algorithm , keeping in mind the progress made by the student as well as the making it easier for the teacher to collect assignments and check for the progress made by the student.

We Also tend to add more commands to the application for due date and progress done which will help the manager , add push notification facility, add alert notification for the manager.

CHAPTER 5

REFERENCES

1. <https://javapoint.com/reference/eclipse/app/Activity>
2. <https://dev.mysql.com/doc/refman/8.0/en/examples.html>
3. <https://eclipse.com/docs/database/jdbc/start/>
4. <https://xampp.com/docs/auth/email-link-auth>
5. <https://sql/docs/auth/php-auth>
6. <https://www.javatpoint.com/eclipse-listview-example>