**ABSTRACT:**

Travel industry is evolving day to day. As the industry evolves the need to digitalize all the transactions becomes need of the hour. This project which is implemented on Java platform helps to manage bus scheduling and bookings. The use of bus traveling is a large growing business in India and other countries. The manual use of bus reservation is presently very strenuous and also consumes a lot of time by having to stay on a long queue. For this reason, an efficient system is to be proposed in this project to ease the issue of bus reservation amongst indigenes within the country. This Bus Booking System is an easily deployable, integrated end-to-end system starting from searching bus routes to book them.

The system is a web – based application that allows visitors to check bus availability, buy and pay bus ticket online. In this paper, the proposed bus reservation system was developed using Hypertext Markup Language (XHTML), JAVA, Structure Query Language (SQL), Spring , Cascading Style Sheet (CSS), AngularJS and JavaScript.

**Keywords:** Bus Reservation, Queue, Efficient

**INTRODUCTION:**

The Online Bus Ticket Reservation System is a web-based application that allows visitors check bus ticket availability, buy bus ticket and pay the bus ticket online

(Asaad, Ayad and Hayder, 2012). This system is established for all the home/office users after gaining access from the administrator. According to Invaderzim (2011), Online Bus Reservation System provides bus transportation system, a facility to reserved seats, cancellation of seats and different types of enquiry which need an instant and quick reservation. This system can be used by the users in performing online reservation via internet for their all business purposes. Users can use this program directly on their websites and no need to install it.

The use of bus traveling is a large growing business in India and other countries; hence bus reservation system deals with maintenance of records of each passenger who had reserved a seat for a journey. It also includes maintenance of information like schedule and details of each bus (Shivaji, 2010). Also, we get to know that there are many operations, which they have to do manually. It takes a lot of time and causes many errors. Due to this, sometimes a lot of problems occur and they were facing many disputes with customers. To solve the above problem, and further maintaining records of items, seat availability for customers, price of per seat, bill generation and other things, we are offering this proposal of reservation system.

However, since the current reservation system is still conducted manually and separately at each branch, contact must be made by each branch’s front-officer to the head office (admin) for each customer’s enquiry in order to get the latest update on schedule, seat availability and other reservation-related information; as well as to avoid duplicate bookings or over-capacity. There is also a physical limit to the reservation availability as each branch only operates during certain hours and reservations can only be made on-the-spot. These limitations are not the only issues the company is currently facing. Other factors that create problems include human errors (e.g. miscalculations in ticket price, mistakes in noting passenger data, etc.), the fluctuation of passengers during certain periods of time that causes a bottleneck in the check-in process because of the inability of the front officer to multitask and the lack of overview or report of the on-going business; making it difficult for the company to judge past/current performance or plan future improvements. Looking at these problems and limitations, it is clear that both the company and the customers require an integrated reservation system that is more efficient in information update and reservation handling and also easy to use. Electronic tickets, or e-tickets, give evidence that their holders have permission to enter a place of entertainment, use a means of transportation, or have access to some internet services. Bus Ticket Reservation System enables the customer to buy bus ticket, make payment, and ask for information online easily. Furthermore, staff can sell bus ticket using Bus Ticket Reservation System after check bus ticket availability for the customer and print the bus ticket to the customer that queue up in the counter. The method to solve this problem is to create an online buying bus ticket system. Customer can buy the bus ticket over the Internet, 24 hours a day, 7 days a week and the bus ticket can't be lost, stolen or left behind. In addition, the online system lets the customers check the availability of the bus ticket before they buy bus ticket (Wee, 2007). Furthermore, customers no need to pay cash to buy bus ticket because they can pay the bus ticket by using deposit slip number order by bank.

**LITERATURE REVIEW:**

**2.1 BUS TICKET SYSTEM**

Bus ticket booking during the offline era posed various difficulties to the customers as well as the bus operators. Offline ticket booking reduced the scope of customers to choose different options based on their travel criterion(Gayathry, 2013). It also increased the franchising cost for the bus operators. At the same time, the bus operators were also finding it difficult to monitor their bus seat filling information. Many small and medium bus service organizations do not have their own online bus ticket booking system. Online Bus ticketing system web portal is a total internet ticketing operations offering the benefit of total in-house management of bus schedules, ticket bookings, ticket sales, report generation, and other business functions associated with ticket sales (Melisa, 2007).It also offers the power of decision making to customers to make a ticket booking through bus operator’s popularity, performance and ranking. This powerful Internet based ticket booking system that allows a full control of not only on the ticketing inventory, but also the site’s content. According to Melisa (2007), stated the basic components of an Online Bus Ticketing System web portal that provides enhanced service to the bus operators and customers consist of the following:

• Capture of customer information such as name, address, phone number and e-mail address

• Price list

• Bus operators ranking

• Seating chart

• Loyalty Points/Redemption

• Search engine

• Payment information

• Organization's advertisement/slogan, phone number, fax number, and address

• Comments and suggestions section / option

• Reports

**2.2 E-TICKET RESERVATION SYSTEM**

E-ticketing could be extended to major entertainment and touristic sites and thus facilitate access to major points of interest within cities, making e-ticketing also interesting for travelers. Urban tourism is the fastest growing tourism sector in the world (Paskaleva, 2014). In public transport, e-ticketing systems are not only means of payment but process huge amount of information which offer a large range of possibilities to make public transport easier to use, to manage and to control. They offer as well opportunities to introduce integrated pricing structure that are not easy to implement with traditional payment tools. Electronic ticketing technologies are classified according to the way they are used for payment. The closer the card is to the payment system, the more reliable the transaction is, but the more constraining it is for the user (Mezghani, 2008). Therefore, the long-term objective is for the customer to be able to pay for public transport without having to show or validate any card, relying on fully automatic fare payment. Public transport operators have been trying to replace paper-based tickets with electronic media, and many countries have implemented or are about to introduce e-ticketing systems. The main characteristic of e-ticketing is that tickets are sold and stored in electronic devices. However, the benefits of a comprehensive e-ticketing system for public transport operators are hard to quantify, as the main aim of e-ticketing is an improved service quality. In monetary terms, e-ticketing could reduce administrative costs as fewer cashiers are needed,

fare processing times could be reduced and a better throughput of passengers could be allowed (Maike, 2014).Moreover, fare evasion and fraud resulting from cash handling could be reduced and better price differentiation would be possible. E-ticketing enables a better integration of alternative services into the scheme, making it more attractive for customers to use it (PricewaterhouseCoopers, 2011). Due to accurate data on passenger flows it might also help to better exploit the network’s capacities and to improve the user experience by setting up tailor-made services for individual passengers. Costs apply that can be easily quantified, e.g., investment and operation costs, particularly the initial one-off costs (e.g., readers, software and consultancy on the scheme design) Integrated schemes appear to be particularly cost intensive, as different applications need to be connected (Wood, Downer, Toberman, 2011).

Additionally, running costs for marketing, maintenance and replacement need to be considered. Costs apply for training staff or resolving passenger disputes and for setting up a (regional or even national) clearing house responsible for centralized data and fare collection. The fear of outsourcing their expertise and responsibilities in ticketing to a third party of suppliers remains a worry to public transport operators (Turner &Wilson, 2010).

**PROBLEM DESCRIPTION:**

**PROBLEM STATEMENT:**

System that are using by the staff at the counter currently is an internal system and just used to sell the bus ticket at the counter. Customer has to go to the counter to buy bus ticket or ask for bus schedule. Furthermore, customers need to pay cash when they buy the bus ticket and sometimes needs to queue up long time to get the bus ticket. Besides that, customer also not allowed to buy bus ticket through telephone and the bus company's telephone always-busy line.

**SOLUTION:**

The method to solve this problem is to create an online booking bus ticket system. Customer canbuy the book ticket over the Internet, 24 hours a day, 7 daysa week and the bus ticket can't be lost, stolen or left behind. In addition, the online system lets the customers check the availability of the bus ticket before they buy bus ticket. Furthermore, customers no need topay cash to buy bus ticket because they can pay the bus ticket on the traveling time.

**ANALYSIS:**

* Requirement analysis is a software engineering task that bridges the gap between the system level software allocation and software design.
* It enables the system engineer to specify software function and performance, indicate software interface with other system elements, and establish design constraints that the software must meet.
* It provides the software designer with a representation of information and function that can be translated to data, architectural and procedural design.

**SYSTEM SPECIFICATION:**

**HARDWARE CONFIGURATION:**

Computer processor : Pentium4 (min)

Hard Disk : 50Gb (min)

RAM : 512MB (min) more

**SOFTWARE SPECIFICATION:**

Operating System : WINDOWS 7 or above

Language used : Java, JSP, CSS, Servlet, HTML, AngularJs, Spring

Data Base : Oracle, Derby

Server : Apache Tomcat 7.0

**4) METHODOLOGY**

**4.1 USE CASE DIAGRAM:**

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. Use cases are nothing but the system functionalities written in an organized manner. The second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications. To draw an use case diagram we should have the following items identified.

* Functionalities to be represented as an use case
* Actors
* Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items one should follow the guidelines to draw an efficient use case diagram.

* The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
* Give a suitable name for actors.
* Show relationships and dependencies clearly in the diagram.
* Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
* Use note whenever required to clarify some important points.

**4.2. CLASS DIAGRAM:**

Class diagrams are the most popular UML diagrams used for construction of software applications. So it is very important to learn the drawing procedure of class diagram. Class diagrams have lot of properties to consider while drawing but here the diagram will be considered from a top level view. Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram:

* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified in advance.
* Responsibility (attributes and methods) of each class should be clearly identified.
* For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
* Use notes whenever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

**4.3. ENTITY-RELATIONSHIP DIAGRAM:**

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. It does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. An ER model can also be expressed in a verbal form, for example: one building may be divided into zero or more apartments, but one apartment can only be located in one building.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a [relational database](https://en.wikipedia.org/wiki/Relational_database) a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the [three schema approach](https://en.wikipedia.org/wiki/Three_schema_approach) to [software engineering](https://en.wikipedia.org/wiki/Software_engineering).

[**CONCEPTUAL DATA MODEL**](https://en.wikipedia.org/wiki/Conceptual_data_model):

This is the highest level ER model in that it contains the least granular detail but establishes the overall scope of what is to be included within the model set. The conceptual ER model normally defines master reference data entities that are commonly used by the organization. Developing an enterprise-wide conceptual ER model is useful to support documenting the [data architecture](https://en.wikipedia.org/wiki/Data_architecture) for an organization.

A conceptual ER model may be used as the foundation for one or more logical datamodels .The purpose of the conceptual ER model is then to establish structural [metadata](https://en.wikipedia.org/wiki/Metadata) commonality for the [master data](https://en.wikipedia.org/wiki/Master_data) entities between the set of logical ER models. The conceptual data model may be used to form commonality relationships between ER models as a basis for data model integration.

[**LOGICAL DATA MODEL**](https://en.wikipedia.org/wiki/Logical_data_model)**:**

A logical ER model does not require a conceptual ER model, especially if the scope of the logical ER model includes only the development of a distinct information system. The logical ER model contains more detail than the conceptual ER model. In addition to master data entities, operational and transactional data entities are now defined. The details of each data entity are developed and the relationships between these data entities are established. The logical ER model is however developed independently of the specific [database management system](https://en.wikipedia.org/wiki/Database_management_system) into which it can be implemented.

[**PHYSICAL DATA MODEL**](https://en.wikipedia.org/wiki/Physical_data_model):

One or more physical ER models may be developed from each logical ER model. The physical ER model is normally developed to be instantiated as a database. Therefore, each physical ER model must contain enough detail to produce a database and each physical ER model is technology dependent since each database management system is somewhat different.

The physical model is normally instantiated in the structural metadata of a database management system as relational database objects such as [database tables](https://en.wikipedia.org/wiki/Database_table), [database indexes](https://en.wikipedia.org/wiki/Database_index) such as [unique key](https://en.wikipedia.org/wiki/Unique_key) indexes, and database constraints such as a [foreign key constraint](https://en.wikipedia.org/wiki/Foreign_key_constraint) or a commonality constraint. The ER model is also normally used to design modifications to the relational database objects and to maintain the structural metadata of the database.

The first stage of [information system](https://en.wikipedia.org/wiki/Information_system) design uses these models during the [requirements analysis](https://en.wikipedia.org/wiki/Requirements_analysis) to describe information needs or the type of [information](https://en.wikipedia.org/wiki/Information) that is to be stored in a [database](https://en.wikipedia.org/wiki/Database). The [data modeling](https://en.wikipedia.org/wiki/Data_modeling) technique can be used to describe any [ontology](https://en.wikipedia.org/wiki/Ontology_(computer_science)) (i.e. an overview and classifications of used terms and their relationships) for a certain [area of interest](https://en.wikipedia.org/wiki/Universe_of_discourse). In the case of the design of an information system that is based on a database, the [conceptual data model](https://en.wikipedia.org/wiki/Conceptual_data_model) is, at a later stage (usually called logical design), mapped to a [logical data model](https://en.wikipedia.org/wiki/Logical_data_model), such as the [relational model](https://en.wikipedia.org/wiki/Relational_model); this in turn is mapped to a physical model during physical design. Both of these phases are referred to as "physical design."

**4.4 HTML:**

**Hypertext Markup Language** (**HTML**) is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for creating [web pages](https://en.wikipedia.org/wiki/Web_page) and [web applications](https://en.wikipedia.org/wiki/Web_application). With [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [JavaScript](https://en.wikipedia.org/wiki/JavaScript) it forms a triad of cornerstone technologies for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).[[1]](https://en.wikipedia.org/wiki/HTML#cite_note-1) [Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [webserver](https://en.wikipedia.org/wiki/Webserver) or from local storage and render them into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/Img_(HTML_element)) and other objects, such as [interactive forms,](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. It provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by *tags*, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img/> and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript) which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

**4.5 CSS:**

**Cascading Style Sheets** (**CSS**) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language). Although most often used to set the visual style of [web pages](https://en.wikipedia.org/wiki/Web_page) and user interfaces written in [HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML), the language can be applied to any [XML](https://en.wikipedia.org/wiki/XML) document, including [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) and [XUL](https://en.wikipedia.org/wiki/XUL), and is applicable to rendering in [speech](https://en.wikipedia.org/wiki/Speech_synthesis), or on other media. Along with HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for [web applications](https://en.wikipedia.org/wiki/Web_applications), and user interfaces for many mobile applications.

CSS is designed primarily to enable [the separation of document content from document presentation](https://en.wikipedia.org/wiki/Separation_of_presentation_and_content), including aspects such as the [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface).[[3]](https://en.wikipedia.org/wiki/Cascading_Style_Sheets#cite_note-3) This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility), provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

**4.6 JSP:**

Architecturally, JSP may be viewed as a high-level [abstraction](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) of [Java servlets](https://en.wikipedia.org/wiki/Java_servlet). JSPs are translated into [servlets](https://en.wikipedia.org/wiki/Java_Servlet) at runtime, therefore JSP is a Servlet; each JSP servlet is cached and re-used until the original JSP is modified.

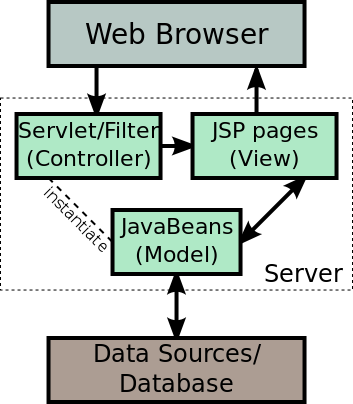
JSP can be used independently or as the view component of a server-side [model–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) design, normally with [JavaBeans](https://en.wikipedia.org/wiki/JavaBeans) as the model and Java servlets (or a framework such as [Apache Struts](https://en.wikipedia.org/wiki/Apache_Struts)) as the controller. This is a type of [Model 2](https://en.wikipedia.org/wiki/JSP_model_2_architecture) architecture.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than [machine code](https://en.wikipedia.org/wiki/Machine_code). Like any other Java program, they must be executed within a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) that interacts with the server's host [operating system](https://en.wikipedia.org/wiki/Operating_system) to provide an abstract, platform-neutral environment.

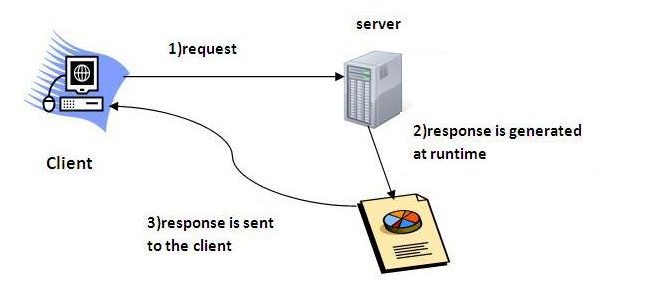
JSPs are usually used to deliver HTML and XML documents, but through the use of Output Stream, they can deliver other types of data as well.[[4]](https://en.wikipedia.org/wiki/JavaServer_Pages#cite_note-4)

The [Web container](https://en.wikipedia.org/wiki/Web_container) creates JSP implicit objects like request, response, session, application, config, page, page Context, out and exception. JSP Engine creates these objects during translation phase.

JSP pages use several delimiters for [scripting](https://en.wikipedia.org/wiki/Server-side_scripting) functions. The most basic is **<% ... %>**, which encloses a JSP scriptlet. A scriptlet is a fragment of Java code that is run when the user requests the page. Other common delimiters include **<%= ... %>** for expressions, where the scriptlet and delimiters are replaced with the result of evaluating the expression, and directives, denoted with **<%@ ... %>**



**4.7 SERVLET:**

****

**Servlet** technology is used to create web application (resides at server side and generates dynamic web page).

**Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language. But there was many disadvantages of this technology. We have discussed these disadvantages below.

There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

Servlet can be described in many ways, depending on the context.

* Servlet is a technology i.e. used to create web application.
* Servlet is an API that provides many interfaces and classes including documentations.
* Servlet is an interface that must be implemented for creating any servlet.
* Servlet is a class that extend the capabilities of the servers and respond to the incoming request. It can respond to any type of requests.
* Servlet is a web component that is deployed on the server to create dynamic web page.

**4.7 SPRING:**

Spring could potentially be a one-stop shop for all your enterprise applications. However, Spring is modular, allowing you to pick and choose which modules are applicable to you, without having to bring in the rest. The following section provides details about all the modules available in Spring Framework.

The Spring Framework provides about 20 modules which can be used based on an application requirement.



**CORE CONTAINER:**

The Core Container consists of the Core, Beans, Context, and Expression Language modules the details of which are as follows −

* The **Core** module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
* The **Bean** module provides Bean Factory, which is a sophisticated implementation of the factory pattern.
* The **Context** module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The Application Context interface is the focal point of the Context module.
* The **SpEL** module provides a powerful expression language for querying and manipulating an object graph at runtime.

**DATA ACCESS/INTEGRATION:**

The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows −

* The **JDBC** module provides a JDBC-abstraction layer that removes the need for tedious JDBC related coding.
* The **ORM** module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
* The **OXM** module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
* The Java Messaging Service **JMS** module contains features for producing and consuming messages.
* The **Transaction** module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

**WEB:**

The Web layer consists of the Web, Web-MVC, Web-Socket, and Web-Portlet modules the details of which are as follows −

* The **Web** module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
* The **Web-MVC** module contains Spring's Model-View-Controller (MVC) implementation for web applications.
* The **Web-Socket** module provides support for Web Socket-based, two-way communication between the client and the server in web applications.
* The **Web-Portlet** module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

**MISCELLANEOUS:**

There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules the details of which are as follows −

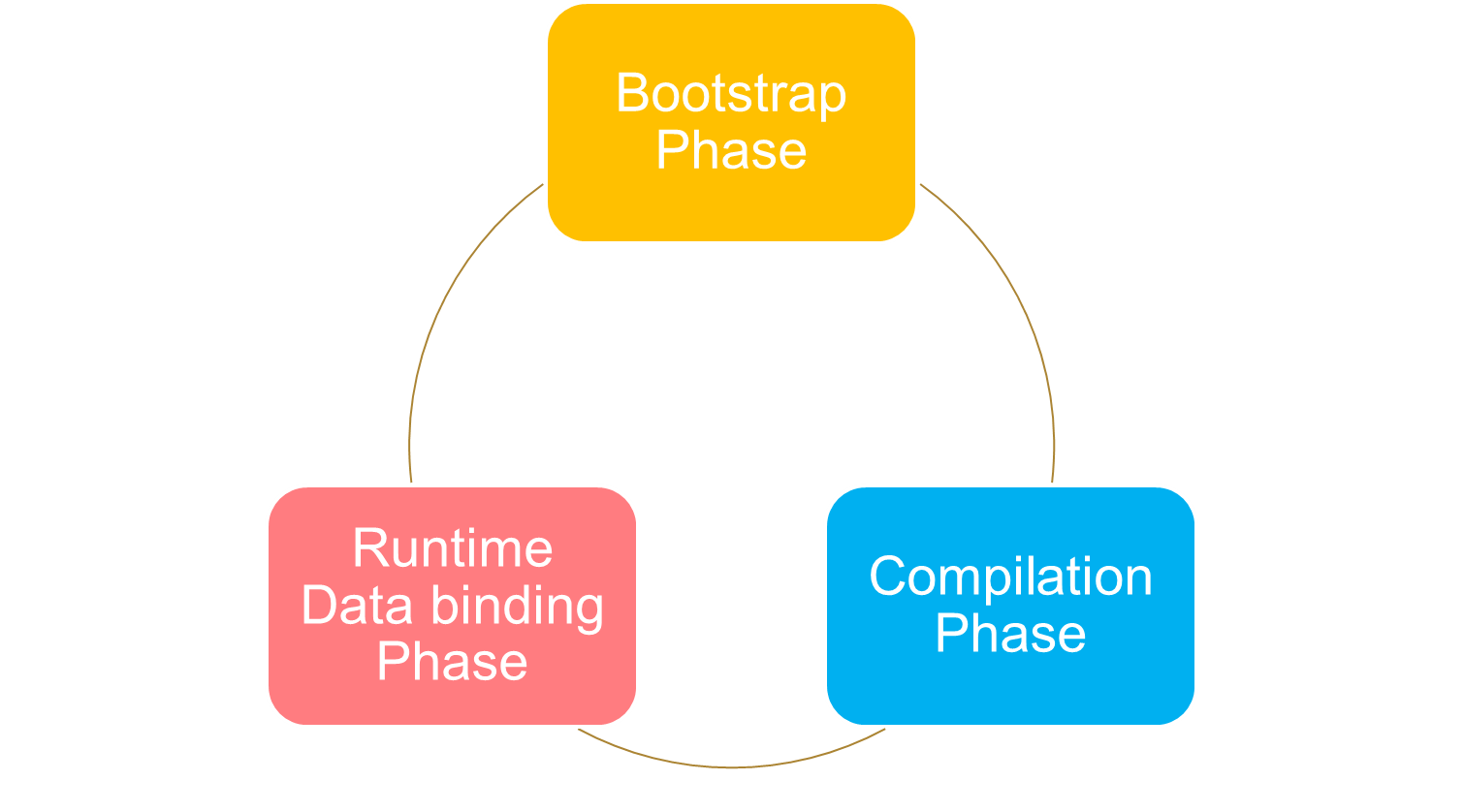
* The **AOP** module provides an aspect-oriented programming implementation allowing you to define method-interceptors and point cuts to cleanly decouple code that implements functionality that should be separated.
* The **Aspects** module provides integration with AspectJ, which is again a powerful and mature AOP framework.
* The **Instrumentation** module provides class instrumentation support and class loader implementations to be used in certain application servers.
* The **Messaging** module provides support for STOMP as the Web Socket sub-protocol to use in applications. It also supports an annotation programming model for routing and processing STOMP messages from Web Socket clients.
* The **Test** module supports the testing of Spring components with JUnit or Test NG frameworks.

**4.8 ANGULARJS**:

Angular JS is an open source JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone. Angular Js is developed by Google. It is an excellent framework for building single phase applications and line of business applications. There are a lot of JavaScript frameworks for building web applications. So, it is a genuine question, why to use Angular JS.

To use Angular JS include the following script in the head tag.

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>**



**Following are the advantages of AngularJS over other JavaScript frameworks:**

* **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
* **Two way data binding:** AngularJS creates a two way data-binding between the select element and the order Prop model. Order Prop is then used as the input for the order by filter.
* **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
* **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.
* Directives, filters, modules, routes etc.

**4.9. HARDWARE REQUIREMENT SPECIFICATION:**

**4.9.1. Client Side:**

The common set of requirements specified by any of the operating systems or any software application is physical computer resource that is hardware. The hardware requirement list is often accompanied by Hardware Compatibility List (HCL), especially in case of operating systems. The HCL lists tested incompatible and compatible hardware devices for a specific operating system or application.

|  |  |
| --- | --- |
| **Processor** | Intel core 2 duo and advance |
| **Speed** | 2.0 GHz |
| **Hard Disk Drive** | 250 GB and above. |
| **Operating System** | Windows, Linux |
| **Memory** | 2 GB RAM and above |
| **System Type** | 32,64 bit Operating System |

Table 4.3.1.1

**4.9.2. Server Side:**

For server an n-tier web based database architecture model is used. A virtual and dedicated hosting is used for server side hosting. The server is up for 24/7. If the server is down, a backup alternate server is used seamlessly.

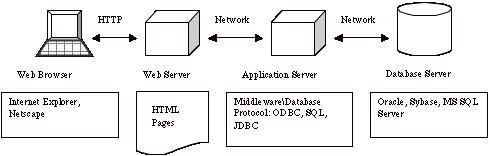


Figure 4.3.2.1

**4.4. SOFTWARE REQUIREMENTS SPECIFICATION:**

**4.4.1) Apache Tomcat Installation**

Apache Tomcat is an open-source Web server and servlet container. It requires a Java Standard Edition Runtime Environment (JRE) version 6 or later. The chit fund system developed uses an Apache Tomcat server version 8.5. Apache Tomcat’s server is called Catalina that contains all the specification for the server that is modified to the needs. Coyote connector component is used to get the response to and fro the server using JK protocol.

**STEPS TO INSTALL:**

1. Download then install the JRE from http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Then download and install the Apache Tomcat, a binary distribution of tomcat from <http://tomcat.apache.org/>
3. Unpack the binary distribution so that it resides in its own     directory (conventionally named "apache-tomcat-[version]").
4. Configure Environment Variables
   * 1. Set CATALINA\_HOME and CATALINA\_BASE (optional).
     2. The CATALINA\_HOME environment variable should be set to the location of the root directory of the "binary" distribution of Tomcat.
     3. The CATALINA\_BASE environment variable specifies location of the root directory of the "active configuration" of Tomcat. It is optional. It defaults to be equal to CATALINA\_HOME.
5. Set JRE\_HOME or JAVA\_HOME.

**5.1.1** The JRE\_HOME variable is used to specify location of a JRE. The JAVA\_HOME variable is used to specify location of a JDK.

**5.1.2** Using JAVA\_HOME provides access to certain additional start-up options that are not allowed when JRE\_HOME is used. If both JRE\_HOME and JAVA\_HOME are specified, JRE\_HOME is used. The best place to include these variables is a "setenv" script.

1. Other Variables like CATALINA\_OPTS are optional to set with. It allows specification of additional options for the java command to start tomcat.
2. Start Tomcat
   * 1. On Windows

                             %CATALINA\_HOME%\bin\startup.bat

                     Or

                      %CATALINA\_HOME%\bin\catalina.bat start

                  Or

                    $CATALINA\_HOME/bin/catalina.sh start

**4.4.2) DERBY Installation**

cdC:\BXP Dev Env\eclipse-kepler-win32-1.0\eclipse-kepler-win32\jdk7-u51\db\bin

set  DERBY\_HOME=C:\Data\jdk7-u51\db;

set  JAVA\_HOME=C:\Data\jdk7-u51\bin;

set  PATH=C:\Data\jdk7-u51\db\bin;C:\Data\jdk7-u51\bin;%PATH%;

set  DERBY\_HOME=C:\Data\jdk7-u51\db

C:\BXP Dev Env\eclipse-kepler-win32-1.0\eclipse-kepler-win32\jdk7-u51\db\bin>startNetworkServer -h  172.00.00.00

**Bringing up your Client :**

cdC:\BXP Dev Env\eclipse-kepler-win32-1.0\eclipse-kepler-win32\jdk7-u51\db\bin

set  DERBY\_HOME=C:\Data\jdk7-u51\db;

set  JAVA\_HOME=C:\Data\jdk7-u51\bin;

set  PATH=C:\Data\jdk7-u51\db\bin;C:\Data\jdk7-u51\bin;%PATH%;

set  DERBY\_HOME=C:\Data\jdk7-u51\db

ij;

**6.2. APACHE MAVEN**

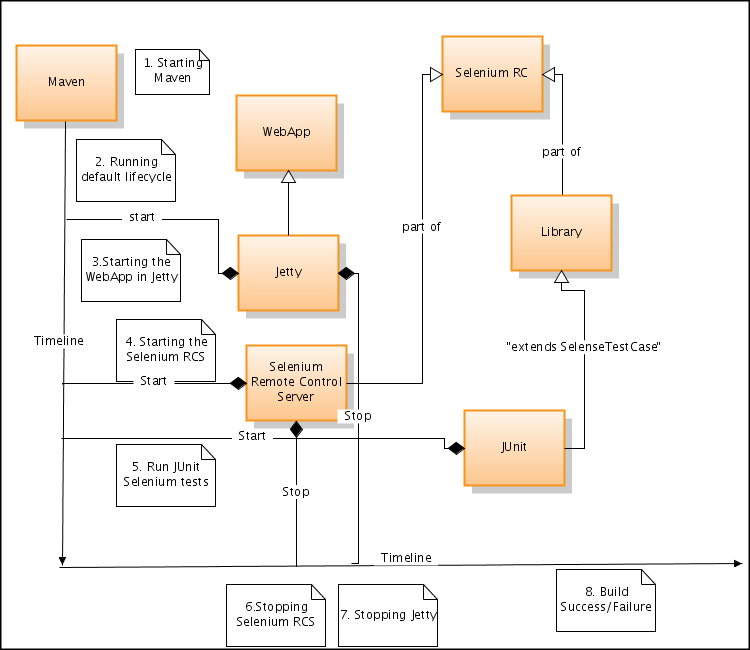
**BLOCK DIAGRAM:**

Figure 6.2.1

Maven is a comprehension and software project management tool that allows a developer to develop based on project object model (POM). Maven’s key feature is dependency management. It helps to download the required JAR’s for the project build.

**DEPENDENCIES**

The core feature of Maven is dependency management. Its dependency handling mechanism is well built up around a coordinate system that identifies the individual artefacts such as software libraries and modules.

A POM provides all the configurations needed for a single project. The general configuration covers the project's name, its dependencies and its owner on other projects. One can also configure the individual phases of the project’s build process that are implemented as plugins.

**PRESENT WORK:**

References:

1. Asad, A.A., Ayad, M.J. and Hayder, N.K. (2012). Design and Developing Online Iraqi Bus Reservation

System Using Unified Modeling Language. *International Journal of Scientific knowledge* Available at:

http://www.ijsk.org/uploads/3/1/1/7/3117743/v3i103\_information\_technology.pdf Accessed 13th December 2014

2. Gayathry, S. (2013). *Online Bus Ticketing System:* Tactful Management Research Journal Through Case Studies.RedBus.in

3. Goessl, L. (2009). *Business Advantages of a Relational Database*. Available at:

http://www.insidebusiness360.com/index.php/business-advantages-of-a-relational-database-15806/

4. Invaderzim08, (2011).*Online Bus Reservation System. Retrieved from*

http://www.studymode.com/essays/Bus-Reservation-System-741084.html Accessed 20th November 2014

5. Maike, J.P. (2014). Train, bus and museum - Interrelations of diverse actors within integrated E - ticketing

schemes. Available at: http://www.mobil-tum.vt.bgu.tum.de/fileadmin/w00bqi/www/Session\_Poster/Puhe.pdf

Accessed 16th October 2014

6. Melisa, K. (2007).*Online Bus Ticketing System:* University Of Malaykuala Lumpur Accessed 17th November 2014

7. Mezghani, M. (2008). Study on Electronic Ticketing in Public Transport. Available at:

http://www.emta.com/IMG/pdf/EMTA-Ticketing.pdf Accessed: 16th November 2014

8. Paskaleva, K. (2014). Integrated Public e-Services: Joining-up Strategies and Technologies for City Available

at: http://www.europarl.europa.eu/RegData/etudes/etudes/join/2014/513551/IPOL

JOIN\_ET%282014%29513551\_EN.pdf Accessed 18th October 2014

9. Pricewaterhouse Coopers (2011). Smart & Integrated Ticketing Report for Scotland. Available at:

http://www.transportscotland.gov.uk/files/documents/reports/j13684/j13684.pdf Accessed: 30th October 2014

10. Shivaji. Varma. (2010). *Bus reservation system. Retrieved from*

http://shivajivarma.wordpress.com/2010/09/20/dbms-project-bus-reservation Accessed 19th November 2014

<https://inautixonline.inautix.com/>– iNautix Company.