Abstract

The concept of a PRISON INFORMATION SYSTEM (PIS) that could maintain details about a prison and its core functionalities by the remission section of the prison was developed after a visit to the prison with the Kansas State University Crime Club. In this system, information is stored and accessed at two levels in order to maintain information security. The existing PIS contains physical copies of all the prisoner and staff information; however, this system is time-consuming, yields error-prone results, delays data retrieval, and lacks accuracy. Therefore, the PIS electronically stores data in a reliable database with proper front end, allowing simple interaction for prison officials. System implementation would increase the method, speed, and accuracy of data retrieval and eliminate the need for officers to spend time storing the prison data.

The system is going to be accessed at two levels: Admin and Jailor. In the Admin level, jailors and jails can be added and all jails, jailors and prisoners can be viewed. At Jailor level, Jailor can access and modify all details pertaining to each prisoner including bail, visitors, medical, and punishment details. This system was developed using java, servlets, JDBC, and JSP for coding and HTML, CSS, and JSP for front-end interface. The database is maintained in MySQL.

Table of Contents

[Chapter 1 - Introduction 1](#_Toc437459402)

[1.1 Problem Description 1](#_Toc437459403)

[1.2 Project Description 1](#_Toc437459404)

[1.3 Methodology to Build the Prison Information System 1](#_Toc437459405)

[1.4 Report Structure 3](#_Toc437459406)

[Chapter 2 - Implementation 4](#_Toc437459407)

[2.1 Tools and Technologies Used 5](#_Toc437459408)

[2.1.1 Interface Tier 5](#_Toc437459409)

[2.1.2 Middle Tier 5](#_Toc437459410)

[2.1.3 Database Tier 6](#_Toc437459411)

[2.2 UML Diagrams 6](#_Toc437459412)

[2.2.1 Use Case Diagram 7](#_Toc437459413)

[2.2.2 Sequence Diagram 9](#_Toc437459414)

[2.2.3 Activity Diagram 11](#_Toc437459415)

[2.3 Database Design 12](#_Toc437459416)

[Chapter 3 - Application GUI and Working 16](#_Toc437459417)

[3.1 Home Page 16](#_Toc437459418)

[3.2 Login Page 17](#_Toc437459419)

[3.3 Admin Services 17](#_Toc437459420)

[3.4 Jailor Services 18](#_Toc437459421)

[Chapter 4 - Evaluation 19](#_Toc437459422)

[4.1 Unit Testing 19](#_Toc437459423)

[4.2 Performance Evaluation 20](#_Toc437459424)

[4.3 User Interface Evaluation 21](#_Toc437459425)

[4.4 Summary 21](#_Toc437459426)

[Chapter 5 - Conclusions and Future Work 23](#_Toc437459427)

[5.1 Conclusion 23](#_Toc437459428)

[5.2 Future Scope 23](#_Toc437459429)

[Bibliography 24](#_Toc437459430)

# Introduction

## 1.1 Problem Description

Each prison must store excessive amount of information regarding prisoners, jailor details, and jail details. This information has previously been stored in manuals, but this method could be replaced with a computerized system that would increase data storage and simplify data retrieval. The proposed system can overcome the cumbersome, time-consuming maintenance of a manual-entry system.

## 1.2 Project Description

The proposed Prison Information System (PIS) was designed to provide a solution for drawbacks of the present system. This system can be operated at two levels: the Admin level and the Jailor level.The Admin module maintains all details regarding the jailors, prisoners, and visitors, and the Jailor module maintains all the information pertaining to the jailors and their interactions related to prisoners, as well as jailor’s personal details and visitor information and visiting times. The Jailor module also contains information about the prisoners, their claims, and their punishment details. The proposed system overcomes disadvantages, such as time consumption and dependency. Use of advanced java concepts such as servlets and JSP also increase project flexibility.

## 1.3 Methodology to Build the Prison Information System

The general methodology to develop PIS included phases that describes the system’s life cycle model for software development in order to build a software divided into fragments for simple tracking and debugging. Phases of PIS before completion are described in the following paragraphs.

**Requirement Analysis Phase**

The initial requirement analysis phase allowed collection of all parameters involved in maintaining a prison and people associate with the prison (i.e., prisoners, jailors, and administrators). This phase clearly indicated expectations of a PIS user.

**Design Phase**

Based on collected requirements, a prototype of PIS was built to cover all requirements of the end user, forming the backbone of the desired interface for that user. This phase built connections between modules such as Jailor, Admin, and Prisoners and identified how they interact together.

**Development Phase**

The development phase included choosing suitable software to fulfill requirements from the design phase. Java was chosen as the core technology, and JSP and servlets modeled the interfaces. JavaScript and AJAX were also used to add additional features like autocomplete to the system. Based on the scale of data used to test the system, MySQL was used as the back-end database to store data.

**Coding Phase**

In the coding phase, software and selected technology specified later in the document were used to build the actual interface based on output from the design phase. JSP pages were primarily used to complete the initial interface, connecting all PIS modules. For features such as searching for prisoners based on name or dynamically updating the prisoner list, JavaScript and AJAX were used to add features for ease of use.

**Testing Phase**

Testing, the last phase of the project, occurred after all necessary modules were built and all the components were put in one place. The entire system was tested to ensure that all requirements gathered initially were satisfied.

## 1.4 Report Structure

This report describes each phase of development of the proposed system. Chapter 2 describes system implementation via software, and Chapter 3 includes application of PIS. Chapter 4 explains how this application was tested and evaluated, and Chapter 5 presents the conclusion and future scope of this application.

# Implementation

## 2.1 Requirements gathering

Requirement gathering involved talking to the staff at prison and some research about the necessity and requirements while maintaining a prison. After lot of web search and help from people working at prisons, the Jailor operations were clear enough as he deals with the prisoners and their activities. But, the real issue raised when deciding about the supervisor of jailors. For this, a user with admin level access was proposed so that he can have access to create jailors and assign them to jails at different locations. Following table summarizes the requirements of end user,

|  |  |
| --- | --- |
| **User Level** | **Requirement** |
| Admin/Jailor | Common login page where user can select his level of access. |
| Admin | Add new jails to the database. |
| Admin | Add new jailors to the database. |
| Admin | View existing jails, jailors and prisoners in the database. |
| Jailor | View all the prisoners list of his prison after logging in. |
| Jailor | Search for any particular prisoner in his prison with suggestions while searching. |
| Jailor | Add/transfer/delete prisoners from the database. |
| Jailor | Add/view/update prisoner’s work details to/from the database. |
| Jailor | Add/view prisoner’s punishment details to/from the database. |
| Jailor | Add/view medical details of prisoner to/from the database. |
| Jailor | Add/view visitor details of prisoner to/from the database. |
| Admin/Jailor | Logout of his account at any point of time. |

## 2.2 Design and Development

An Apache tomcat server with three-tier architecture1 allowed the PIS to be easily built and integrated. The top tier takes care of the interface, which was HTML in JSP pages; the middle tier involved the logic, which used Java, JSP, and JavaScript; and the database tier stored all the prison information using MySQL database. Tomcat v7.0 was used to integrate all three tiers on the server in order to successfully run the system. Use of this server was advantageous because any tier was easily modified or replaced without affecting the other tiers. In addition, separation of the application and database functionality resulted in decently good load balancing. This server also provided high performance, lightweight, persistent objects and a high degree of flexibility in deployment platform and configuration.

### 2.2.1 Interface Tier

The interface, or top tier, of the architecture involved modelling the interface primarily using HTML embedded with JSP pages. CSS was also used to give the necessary look and feel for all developed JSP pages2. Because JSP was used, overhead associated with HTML was removed since all HTML code was included in the JSP pages. Eclipse mars IDE was used to build the Java model and JSP pages, including external servlets library.

### 2.2.2 Middle Tier

Java 1.7 JDK was used to model connections between the interface and database using a JDBC connection. JDBC drivers were installed prior to using the connection string. Java was the primary coding language because of the ease it provides with several tools and other scripting languages, such as JavaScript3 and AJAX.

The present PIS system required to design web pages which would perform database operation in tandem. Features provided by JSP were used in PIS to satisfy this necessity. JSP pages were created in which HTML tags were used to design the front end while executing database commands on the same page. JSP also allows convenient navigation from one page to another with minimum overhead if JavaScript or jQuery must be used. Approximately 45 JSP pages were created to complete the system with all specified operations. In addition, the AutoComplete feature provided by AJAX was used to offer search suggestions to the user.

### 2.2.3 Database Tier

The database tier modelled the database into categories, or tables, based on the initial requirement. The MySQL4 database was used to store all system data. However, in order to use the JDBC API with a particular database management system, a JDBC technology-based driver must be used to mediate between JDBC technology and the database. Therefore, the MySQL-JDBC connector with JDBC drivers was installed beforehand in order to connect to the database. MySQL was used because it is one of the most popular relational database management systems and it is free and available on almost all platforms.

## 2.3 UML Diagrams

An entire system can be easily interpreted when shown pictorially. Use case diagrams illustrate the actors and processes in PIS. Actors are users that access PIS, and processes are the operations the users can perform5 Sequence diagrams illustrate how objects in the system interact with each other and the order in which those interactions occur5 The activity diagram in Figure 2.6 shows the operational work flow within PIS (i.e., how a user can flow from one page to another), similar to a flow diagram5.

### 2.3.1 Use Case Diagram

The use case diagram in Figure 2.1 shows activities that a user with admin access can perform in PIS. The administrator is the actor, and the processes are operations that the administrator can perform. The administrator can perform operations such as adding jails and jailors or viewing jails, jailors, prisoners, punishments, and visitors. Each process in this diagram typically represents each web page a user can view.

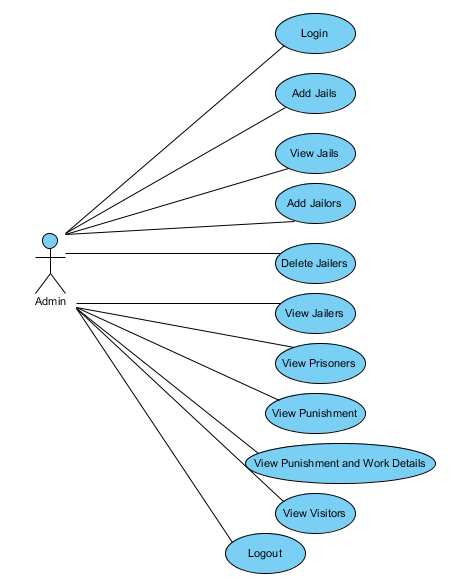


Figure .1 Use case diagram for administrator

The use case diagram in Figure 2.2 shows activities that a user with jailor access can perform in PIS. Jailor is the actor, and the processes are operations that the jailor can perform, such as adding prisoners, transferring prisoners, and viewing prisoners, punishments, bails, disposals, and medical reports. Each process in this diagram typically represents each web page a user can view.

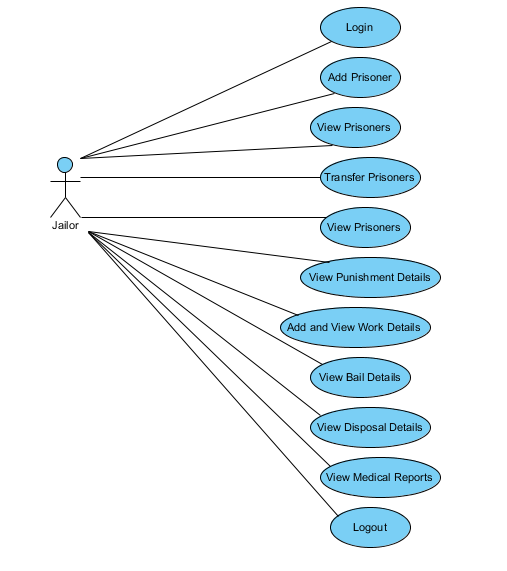


Figure .2 Use case diagram for jailor

### 2.3.2 Sequence Diagram

A sequence diagram illustrates how groups of objects collaborate in some behavior over time, capturing the behavior of a single-use case and showing the objects and messages that are passed between objects in the use case. Figure 2.3 shows the sequence of how objects are connected within the system when the jailor is accessing PIS. As shown in the figure, the jailor initially logs into the system and then performs operations such as viewing, adding, or deleting prisoner details. This sequence diagram shows how a jailor can enter credentials, log in as jailor, and then perform multiple operations from there like viewing prisoners, punishments, bail details, etc.

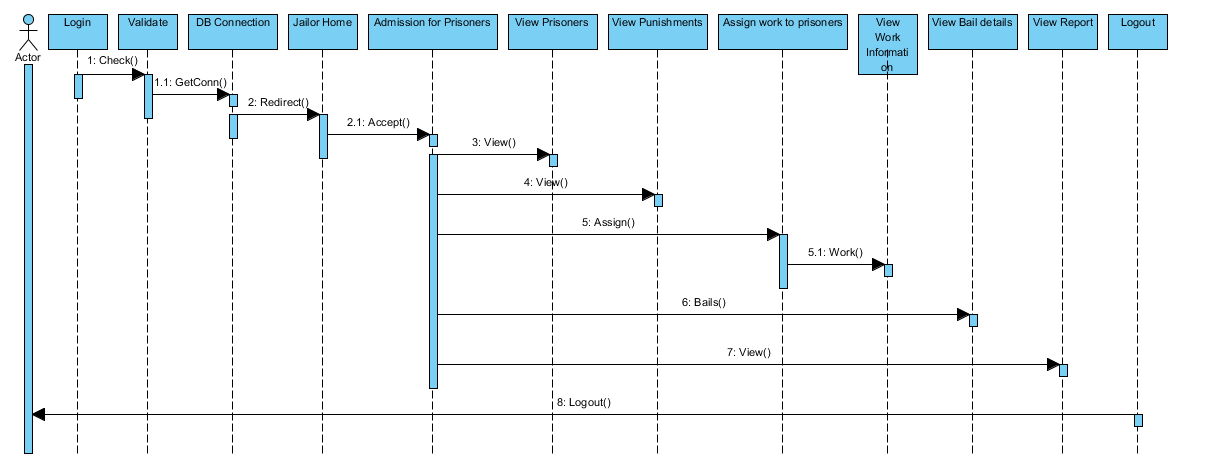


Figure .3 Sequence diagram for jailor login

Figures 2.4 and 2.5 show interactions between objects when an administrator accesses PIS. The actor “Admin” can access the system, thereby accessing the database. Admin can then perform operations such as adding jailors and viewing prisoner details, which involve actor to system interaction. The system then interacts with the database to verify the user credentials or to enter jailor details. These operations are performed in the back end.

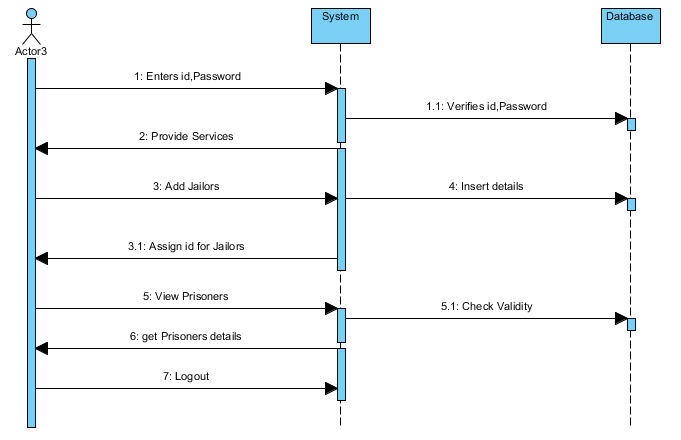


Figure .4 Sequence diagram for services for an administrator

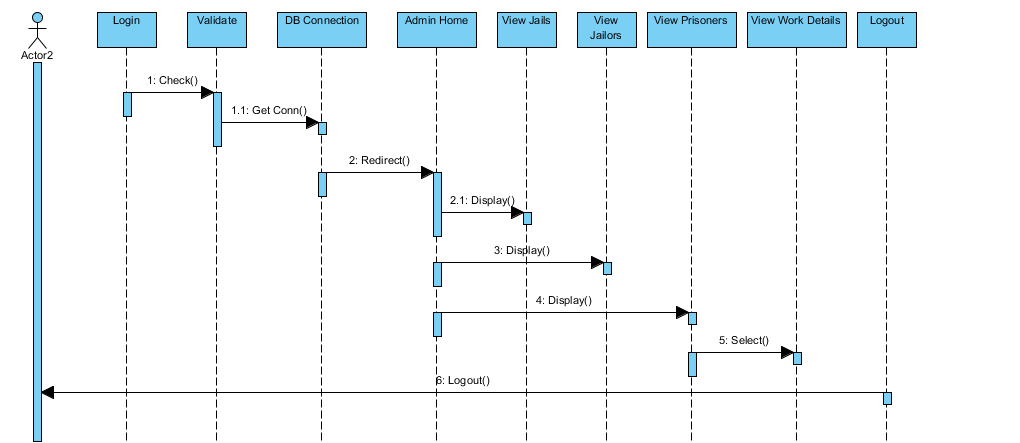


Figure .5 Sequence diagram for services for an administrator

### 2.3.3 Activity Diagram

Activity diagrams describe the workflow behavior of a system and the state of activities by showing the sequence of activities performed. Activity diagrams can show conditional or parallel activities. The activity diagram in Figure 2.6 represents each operation that can be performed by jailors, administrators, and common operations, such as logging in and out of the system. Operations such as adding and deleting jails for the administrator or adding and deleting prisoners for the jailor are represented in the figure.

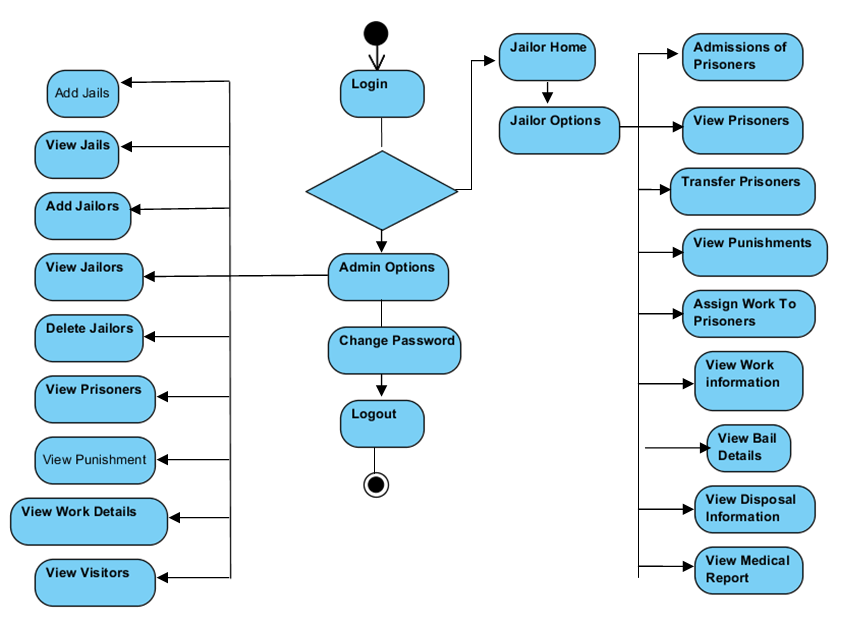


Figure .6 Activity diagram

## 2.4 Database Design

Optimal database6 design for any web application is as important as the user interface and logical designing. Logical designing initially involves coding the middle tier logic to satisfy client needs. The latter part of logical designing redirects to database design, which should be able to hold all necessary data without redundancy and valid constraints. In PIS, the MySQL database was used to store the data, and the subsequent database was designed in MySQL. Based on requirements gathered, data pertaining to user credentials for system login, jailor details, jail details, and prisoner data such as names, punishment details, and work details must be stored in PIS. Therefore, in order to prevent redundant data columns, 11 tables were created that were related using primary keys and foreign keys to store all data.

Table 2.1 verifies credentials and stores credentials of PIS users with roles of Admin or Jailor. The column *userid* stores the user’s login ID; the respective password is shown in the *password* column. Only the administrator can add new rows to this table from the front end.

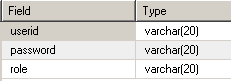
**Table 2.1 Login**

Table 2.2 stores the jailor’s information and relevant jail details assigned to the jailor by the administrator. The jailor data can contain information regarding the jailor’s address, gender, phone, and allotted jail name, all stored in individual columns. The administrator can create new jailors (i.e., add new rows to this table).

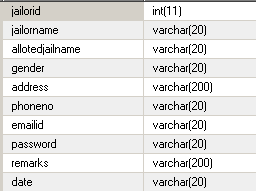
**Table 2.2 Jailor’s registration**

Table 2.3 stores information regarding the jail and jails location. Jails stored in this table are assigned to jailors by the administrator. The jail creation and jail assignment in the database can only be made by the administrator and are updated or created in this table.

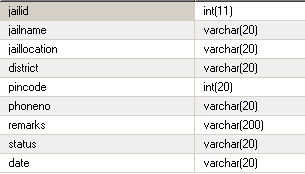
**Table 2.3 Jail registration**

Table 2.4, which stores all prisoner information from all the jails, can be separated for each jail using *jailid*. This table is accessed whenever a jailor or administrator views prisoner information, but only a jailor can add data to this table. This table can also be accessed to autocomplete the prisoner’s name when the user is searching for a particular prisoner.

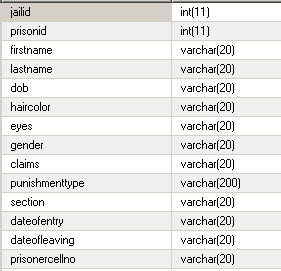
**Table 2.4 Prisoner’s registration**

Table 2.5, which lists a prisoner’s work, is related to a prisoner with *prisonid*. This table is accessed when a user with jailor access views the work details of a particular prisoner. This table can be modified only by users with jailor access.

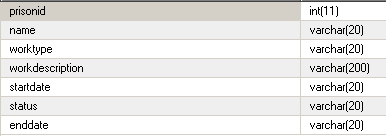
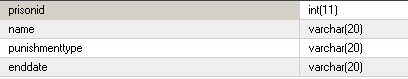
**Table 2.5 Work details**

Table 2.6, which stores details about a prisoner’s punishment, is related to a prisoner by *prisonid*. Similar tables, such as *baildetails*, *medicalreports*, *transferdetails*, *visitors*, and *disposaldetails*, belong to one genre and can be accessed or modified only by users with jailor access. These tables contain information related to particular prisoners using the primary key *prisonid*.

**Table 2.6 Punishment details**

# Application GUI and Working

The final PIS was a website that can be accessed by users with appropriate credentials. This website directs users to various modules through hyperlinks and can be replaced with the legacy system such as excel records or physical records. PIS can be deployed on any server and can be made available at the user’s desktop computer without installation overhead.

## 3.1 Home Page

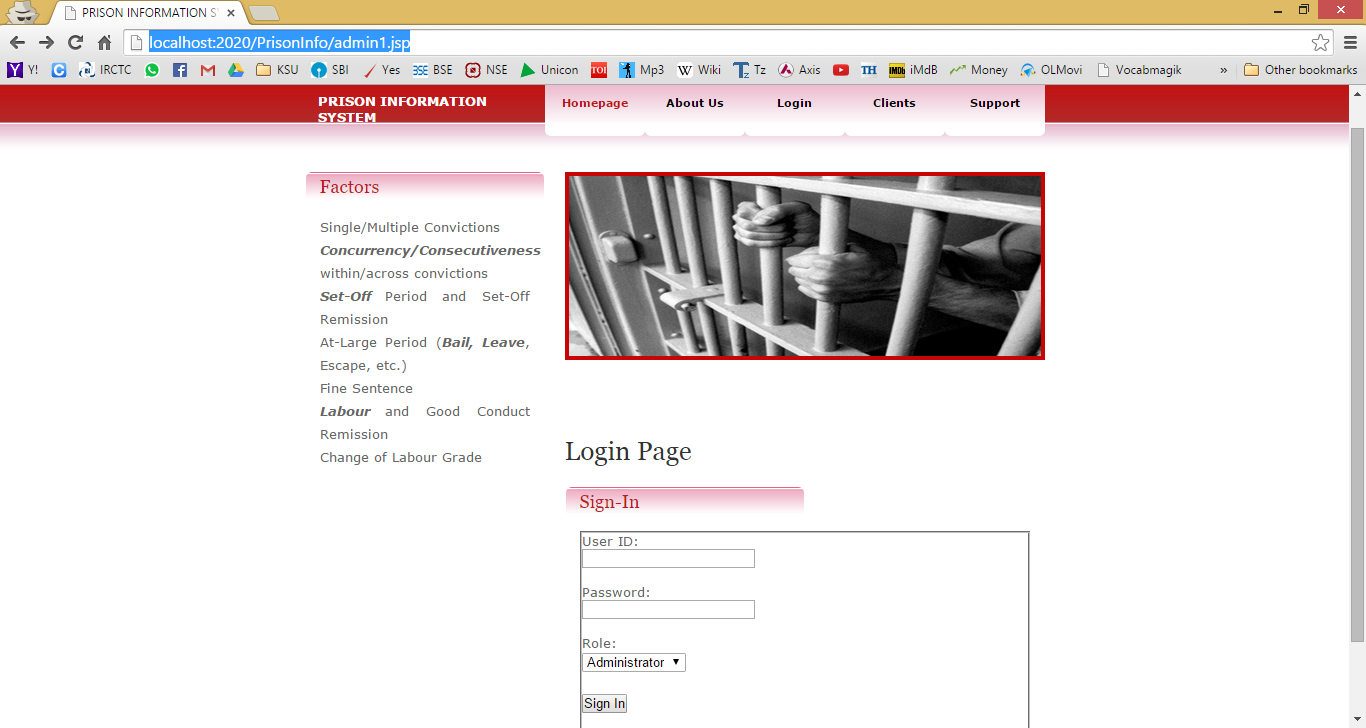
Figure 3.1 shows the PIS home page with options such as About Us, Login, Clients, and Support, offering basic information about the PIS. This home page which is viewable at any level of access (jailor or administrator).



Figure 3.1 Home page of PIS

## 3.2 Login Page

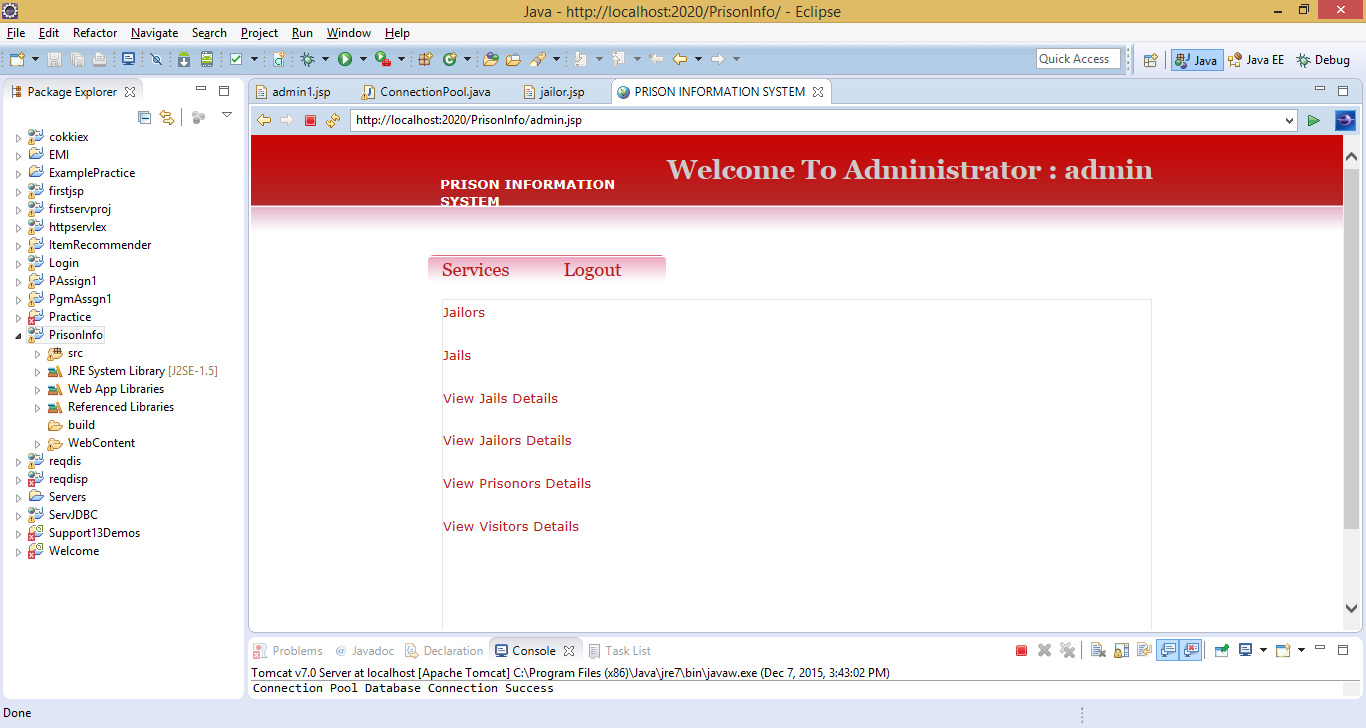
The login page, shown in Figure 3.2 and accessed directly from the PIS home page, allows a user to log in by entering his or her credentials and role (administrator/jailor). Similar to the home page, this page is viewable to all users regardless of access level. This page validates user credentials and alerts the user if the credentials are incorrect.



**Figure 3.2 Login page**

## 3.3 Admin Services

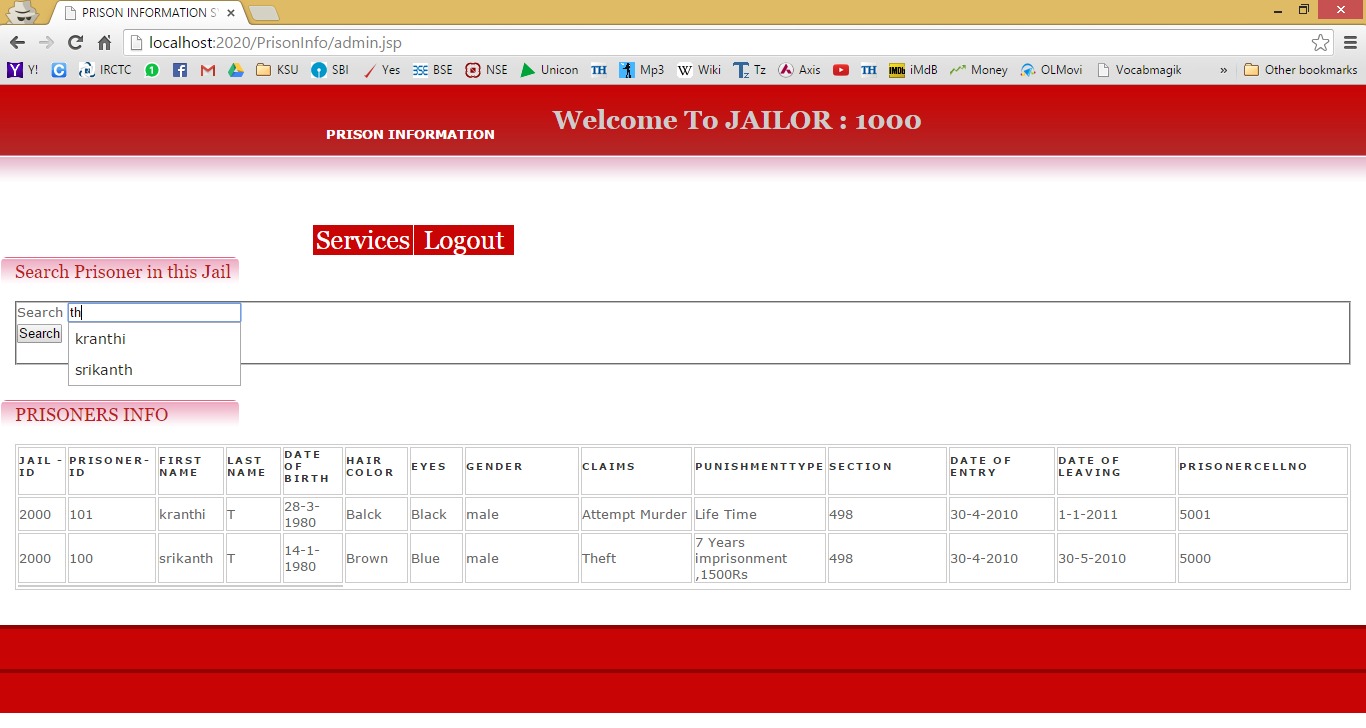
The admin services page shown in Figure 3.3 is visible only to users with administrator access. Operations to be performed are listed and a logout link is included. An administrator has access over jailors, Jails, allowing operations such as adding, deleting, and viewing them. All administrator services are listed under the Services link, which is visible when the mouse pointer hovers over it.



**Figure 3.3 Admin services**

## 3.4 Jailor Services

The jailor services page is visible when a user selects the *Jailor* role and provides appropriate credentials, allowing the jailor to view a list of all prisoners in the jail. The jailor can then search for a particular prisoner using the search bar with the AutoComplete feature and then perform operations such as adding and updating prisoner details by hovering over the Services hyperlink.



**Figure 3.4 Jailor services**

# Evaluation

In the software development process, errors can be injected at any stage during development. Therefore, software testing, which represents the ultimate review of specification, design, and coding, is a critical element of software quality assurance. Testing presents an interesting anomaly for the software engineer. The main objective of testing PIS was to uncover any unusual activity the user was not expecting. Testing is typically done via functional testing, structural testing, or unit testing. Each testing methodology has its own significance to find bugs in the system; however, because of the small scale on which the PIS system was built, unit testing was used to evaluate PIS by setting few parameters and ensuring that all those are satisfying to conclude that the system works as intended.

## 4.1 Unit Testing

Unit testing individually tests each part of the application. Table 4.1 shows the unit tests that were manually performed for this research.

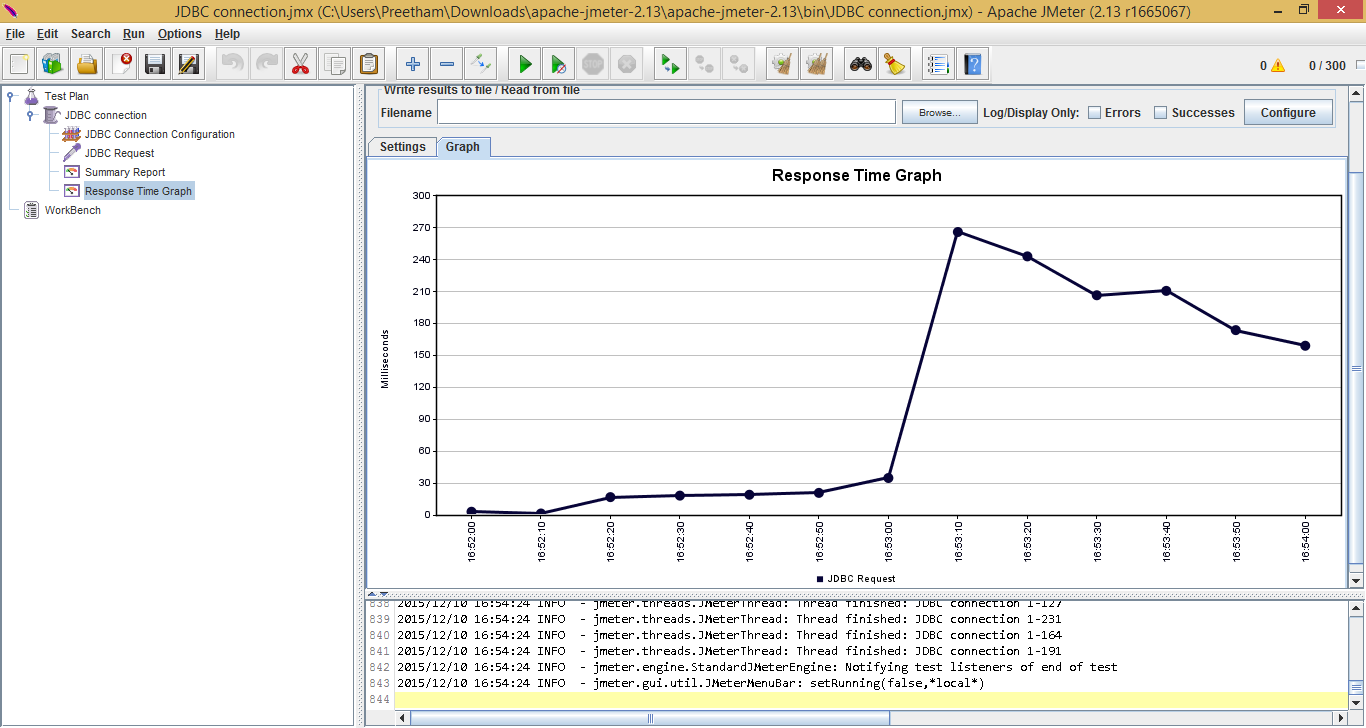
**Table 4.1 Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Test case** | **Expected Result** | **Result** |
| 1. | Open home page | Home page should open. | Pass |
| 2. | Open login page | Login page should open. | Pass |
| 3. | Should alert incorrect credentials | Display that credentials are incorrect. | Pass |
| 4. | Administrator login | Administrator should log in when appropriate role is selected. | Pass |
| 5. | Jailor login | Jailor should login when appropriate role is selected and view all the prisoners in that jail. | Pass |
| 6. | Administrator should be able to add jailors | Jailor should be added to database. | Pass |
| 7. | Administrator should be able to add jails | Jail should be assigned to database and jailor should be assigned to it. | Pass |
| 8. | Administrator should be able to view jailor/jail/prisoner details | All details in the database should be displayed. | Pass |
| 9. | Jailor should be able to view all services when hovered over it | All operations should be enabled when hovered over Services hyperlink. | Pass |
| 10. | Jailor should be able to search for prisoners in the jail | Jailor should receive suggestion when searching for prisoners. | Pass |
| 11. | Jailor should be able to add work/ medical/ visitor/ disposal details | Appropriate details should be added to the database. | Pass |
| 12. | Jailor should be able to view work/ medical/ visitor/ disposal details | Appropriate details should be retrieved from the database. | Pass |
| 13. | Jailor/administrator should be able to logout | User should be directed to login page after logging out. | Pass |

## 4.2 Performance Evaluation

Since PIS is divided into three sections, communication between the sections becomes a major issue when it comes to performance evaluation. Communication between the interface and database can be evaluated using the Apache JMeter. This tool tests the JDBC connection with sample queries. A test conducted on a Windows 8.1 system with i5 processor and 6 GB RAM with sample data yields the following results.

This sample has 100000 records and varying number of users to check for maximum number of users the server can accommodate at a particular interval of time. Starting from 100 users, gradual increase of number of users until 300 users generated the graph shown in Figure 4.1. This shows that the system works until 220-230 users use PIS database concurrently. This performance would increase if a high performance server is used.



**Figure 4.1 Response time graph**

## 4.3 User Interface Evaluation

Several CSS templates were tried before finalizing the template used in PIS. The author’s peers and supervising professor were polled before finalizing the CSS template from [www.freetemplates.org](http://www.freetemplates.org). This interface provides simple navigation between pages and looks attractive.

## 4.4 Summary

The objective of this project was to build a dynamic system with which jail staff could efficiently manage people. Although the functionality of the developed system was as expected, the look and feel did not meet expectations; improvements could be made with advanced CSS templates. Because of the technology used, the utilized CSS template was a compromise. You get to learn jQuery, AJAX and their integration with JSP while building this project. Table 4.2 shows the programming/scripting languages used for this system.

**Table 4.2 Technologies Used**

|  |  |
| --- | --- |
| **Language** | **Lines of Code (LOC) Approx.** |
| Java | 100 |
| JavaScript | 100 |
| JSP | 70 Pages (30 Lines) |
| HTML | 1000 |
| AJAX | 200 |
| CSS | Template |

# Conclusions and Future Work

## 5.1 Conclusion

The proposed user-friendly PIS added several enhancements to the currently utilized system. This PIS allows most jailers and administrators to save time by using it. PIS also provides features such as transfer prisoners, view work details, view prisoner details, and access of multiple prisoners in a single login. However, features of this system can be further enhanced.

## 5.2 Future Scope

PIS could be improved by adding features such as automating admin services, resulting in continuous flow of records from the database. Jail information could also be automated. PIS could be further improved by extending the functionality and implementation to mobile devices and other platforms.

Bibliography

[1] Ali Tarhini, (2012, Jan), Concepts of three-tier architecture, [Online], Available: <https://alitarhini.wordpress.com/2011/01/22/concepts-of-three-tier-architecture/>

[2] JSP Tutorials, [Online], Available: [www.roseindia.net/jsp/jsp.htm](http://www.roseindia.net/jsp/jsp.htm)

[3] JavaScript forms, [Online], Available: <http://www.w3schools.com/js/>js\_validation.asp

[4] MySQL Reference Manual, [Online], Available: <http://dev.mysql.com/doc/refman/5.7/en/>

[5] UML-Standard Diagrams, [Online], Available: <http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm>

[6] DBMS-Normalization, [Online], Available: <http://www.tutorialspoint.com/dbms/database_normalization.htm>

[7] Sivateja Kandula, (2012, Sep), Get AutoComplete feature, [Online], Available: <http://www.java4s.com/jquery-tutorials/example-get-autocomplete-feature-in-javajsp-with-jquery-api/#comment-4315>