Prison Management System

Project Report submitted in partial fulfilment of the requirement

for the degree of

Bachelor of Technology

in

Computer Science & Engineering

Under the Supervision of

Dr. Deepak Dahiya

By

Nitika Negi (111237)

to

Jaypee University of Information and Technology

Waknaghat, Distt Solan- 173234 (HP)

Certificate

This is to certify that project report entitled "Prison Management System", submitted by Nitika Negi in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: **Supervisor's Name**

Designation

Acknowledgement

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this B.Tech project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and inspiring views on a number of issues related to the project.

I am especially grateful to Dr. Deepak Dahiya , Project Supervisor , for his valuable suggestions, support and constant encouragement during the course of the project.

Date: **Nitika Negi**

(111237)

Table of Contents

List of Figures

Abstract

This project titled “Prison Management System” aims at providing management software for handling the inner working of Tihar Jail across the country. Since Tihar Jail already offers certain recreational facilities, employment opportunities and educational prospects with an official degree, however on my research, I realized that there was no inner management system for handling all activities taking place within the prison premises and could use a lot of improvisation. The report is based on 5 main sections- Introduction and Motivation, Literature Review, Design Modules, Implementation, Benefits and Scalability. Introduction and Motivation deals with the details of facilities offered with aim of transformation of prisoners so that they don’t resort to their criminal ways, it also elaborates the inspiration taken from Kiran Bedi, former IPS who had a vision of transforming criminals into civilized human beings. The next section- Literature Review contains details of 2 papers and 1 case study I went through before creating design structure for my project. The two papers were one of Texas Prison System and Mumbai Prison System for Arthur Jail Road for have a grasp of both national and international facilities regarding the subject. PRISMS- the Prison Management System for Goa was the case study I went through since it received rave reviews since its establishment. The third section entails the four main design modules for my project namely- Types of Tasks, Sentence Serving, Parole and Interview Handling, and Complaint Handling. It also contains how each of these modules will be implemented and what features they offer. The four section deals with the implementation of the project and the tools used. The fifth section shows the benefits of the software and how it will ease the managing of activities within the jail premises. The last section is about how the project can be expanded to include additional features, new methodologies, and how it can be modified to different jail systems all across the country.

Chapter 1- Introduction and Motivation

1.1 History

Originally, Tihar was a maximum security prison run by State of Punjab. In 1966, control was transferred to National Capital Territory of Delhi. Beginning in 1984, additional facilities were constructed, and the complex became Tihar Prison.

Under the charge of Kiran Bedi, when she was Inspector General of Prisons, she instituted a number of prison reforms at Tihar, including changing its name to Tihar Ashram. She also instituted a Vipassana meditation program for both staff and inmates; initial classes were taught by S. N. Goenka. The Prison has also produced an inmate who has passed the Indian Administrative Service civil service examinations.

Many of the inmates continue their higher education through distance education. The campus placement programme was launched in 2011 for the rehabilitation of inmates about to complete their sentences. In 2014, a recruitment drive led to 66 inmates selected on the basis of their good conduct, received job offers with salaries up to ₹35000 (US$570) per month, from as many as 31 recruiters, which included educational institutions, NGOs and private companies.

1.2 About

Jails all over the world have transformed into a place of reformation. In India, the Tihar Jails have provided various facilities such as recreational facilities, educational facilities, panchayat systems, psychological treatment, yoga and meditation, creative art therapy to help overall development of inmates. Various bakery products, shirts, handlooms, paintings, etc. are available online under the brand name “TJ’s” for people to purchase. These products are designed and manufactured by the inmates. Also a food court is established under the name “Tihar Food Court”, which has received rave reviews for the quality of food and gesture of inmates.

So my project aims to create a management system for record management and automated task allocation for efficient time utilization, cost minimization and to help the government convert convicts into civilized individuals.

The project serves three main objectives:

- To deliver a system that covers all aspects of prison management from the admission to the release of the prisoner and connectivity across jails; brings in administrative efficiency and security; leads to prisoner empowerment.
- Using technology to bring transparency in the system and provide precise implementation of rules and make data easily available to promote efficient decision making.
- To bring in maximum accuracy in the prison management in all key functional and operational areas; to overcome the manual system in the process with efficient record keeping and make the work profile of staff much easier and smooth in day-to-day administrative work.

1.3 Motivation

Kiran Bedi, national icon, and former IPS has a vision that criminals should be involved in self-improvement so that after serving their sentence, they can be a part of society once again and do not resort to their ways. This led to reformation of Tihar Jail, to Tihar Ashram. I would further like to contribute to the system by providing my software to the Government of India to help improve this initiation. In my belief, technology's primary purpose is to reach out all sections of society and help raise their standard of living.

Chapter 2- Literature Review

2.1 Paper 1- ***Prison Models and Prison Management Models and the Texas Prison System***

Submitted By- Gevana Lynn Salinas (Texas State University- San Marcos, Dept. Of Political Science, Public Administration Program), Summer 2009

Objective- To identify the characteristics of the Texas Prison System as compared to previous hierarchical and differentiated model with the model Texas Prison System is currently using and hence form a new module.

Abstract- Purpose: The first purpose of this study is to describe the characteristics of the two prevailing prison system models and three prison management models through the use of scholarly literature. The second purpose is to conduct a research study to describe which prison system model and prison management model the Texas Department of Criminal Justice (TDCJ) is using to both operate and manage the Texas Prison System. Finally, the study will present conclusions and recommendations for future research.

Methodology: The methodology used in this research study is document analysis. Documents were reviewed and retrieved from agency and division mission statements; agency and division overviews; agency budgets; an agency survey; and policy and procedure handbooks and manuals. These documents were used to implement the conceptual framework.

Results: The results showed the TDCJ shares characteristics from the Hierarchical and Differentiated Model, as well as the Control and Responsibility Model. However, the TDCJ appears to operate under the Differentiated Model and is managed under the Control Model based on the research. The mission statements, division overviews, and policy and procedure explain the primary goal of the Texas Prison System is to rehabilitate and reintegrate offenders back into society as productive law-abiding citizens. The Texas Prison System and prison administrator's primary goals are to maintain control of the prison system and the care, custody, and control of inmates following strict guidelines and policy and procedure.

Contribution-

- Inclusion of fair division of labour
- Efficient resource allocation
- Inclusion of professionals, Psychologists, Doctors, Social Workers
- Interest group philosophies
- Re-location of ex-offenders

2.2 Paper 2- *Prison Management System for Arthur Road Jail*

Submitted By- Submitted by Rahul Singh, Sr. Consultant, KPMG, 2010

Objective- To keep record of inmates, establishment/administrative functions, visitors management, record of work allocation, record person’s personal belongings.

Abstract- Prison Department, Government of Maharashtra has implemented Prison Management System (PMS) at Mumbai Central Prison and Mumbai District Prison, in the year 2010 on pilot basis, with a vision to scale up PMS to all prisons across Maharashtra. However, due to certain challenges the department is in the process of evaluation of different options/solutions available to computerize different prisons across Maharashtra. This case study provides experiences/learning from the pilot project, snapshot of various available PMS options/solutions and brief on the way forward for implementation of PMS at various other prisons across Maharashtra.

Contribution-

- A client server based system deployed at the prison premise with no/limited access to internet due to security reasons.
- KBS Computer Private Limited has provided at the all hardware components such as servers, desktops, multifunctional printers, system software licenses etc. at present the contract of M/s KBS Computer Private Limited is nonexistent hence no hardware support is provided.
- The PMS application is developed by AGL, and automated various functions such as record of prisoner right from entry to exit, establishment section, lab, pharmacy, OPD, canteen ration and cloth management. However it may be inferred that *automation of existing process has been done without considering process reengineering prior to automation of the process/sub-process*. As per the contract the IPR is owned by M/s AGL and the annual maintenance contract was recently renewed and there was a period when no support was provided by AGL.

2.3 Case Study 1- *Prison Management System (PRISMS) e-Governance Project of the Govt. of Goa*

Submitted By- Osama Manzar, Social Entrepreneur, December 2012

Objective- To analyze PRISMS, implemented in jails and judicial lock-ups all over Goa by the Office of the Inspector General of Prisons.

Abstract- The Prisons Management Systems (PRISMS), a landmark e-governance initiative of Goa government has not only been pioneering but also influential in many such initiatives in other states as far as automation and efficiency of Prisons management is concerned. PRISMS is an effective ICT-enabled prison administration and management system with the objective of providing an easy effective and efficient mechanism benefiting the prisoners and the concerned prison department. This case study aims to bring forth the importance of Government Process reengineering (GPR) in the context of implementing ICT systems and the resultant benefits to stakeholders.

The period prior to implementation of PRISMS was marked by multiple complexities and hurdles including manual based time consuming process, human errors, insufficient security due to time consuming record keeping, difficulty in managing visitors, faulty calculation of correct remission and release dates, delay in application process, negligence of records and so on. The shortcomings fostered corrupt and inefficient administration and compromised constitutional rights of prisoners and the rule of law.

Post PRISMS, implementation has weeded out key limitations of prisons management in Goa resulting in drastic improvements in prison administration and in the lives of the prisoners. Whereas PRISMS has emerged as a source of tremendous positive change among the stakeholders, it has also established its sustainability on the basis of being cost-effective.

However, PRISMS has had its own share of challenges and limitations; for example, difficulty in motivating the staff in the new system, system design and deployment with 23 diverse modules, network building, and ensuring system foolproof and monitoring.

The objective of the case study is to analyze PRISMS, implemented in jails and judicial lock-ups all over Goa by the Office of the Inspector General of Prisons. The analysis is divided into 5 major sections and focuses on a detailed analysis of the pre-implementation state of the project, the current status with respect to the proper functioning and benefits provided by the system, challenges and lessons learnt during the implementation phase. In terms of methodology, secondary as well as primary data was collected, including interviews of representative stakeholders, namely the office of the Inspector General of Prisons, GEL – the agency that designed and implemented the solution, the prison management staff, and the officers working with the system as well as the prisoners. The findings arrived through this case study point towards the positive impact of e-governance programs with regard to integrated and holistic prison management.

Methodology-

- Secondary and primary data collected from interviews with various stakeholders.
- The findings arrived through this case study point towards the positive impact of e-governance programs with regard to integrated and holistic prison management.

Chapter 3- Design Modules

3.1 Outline

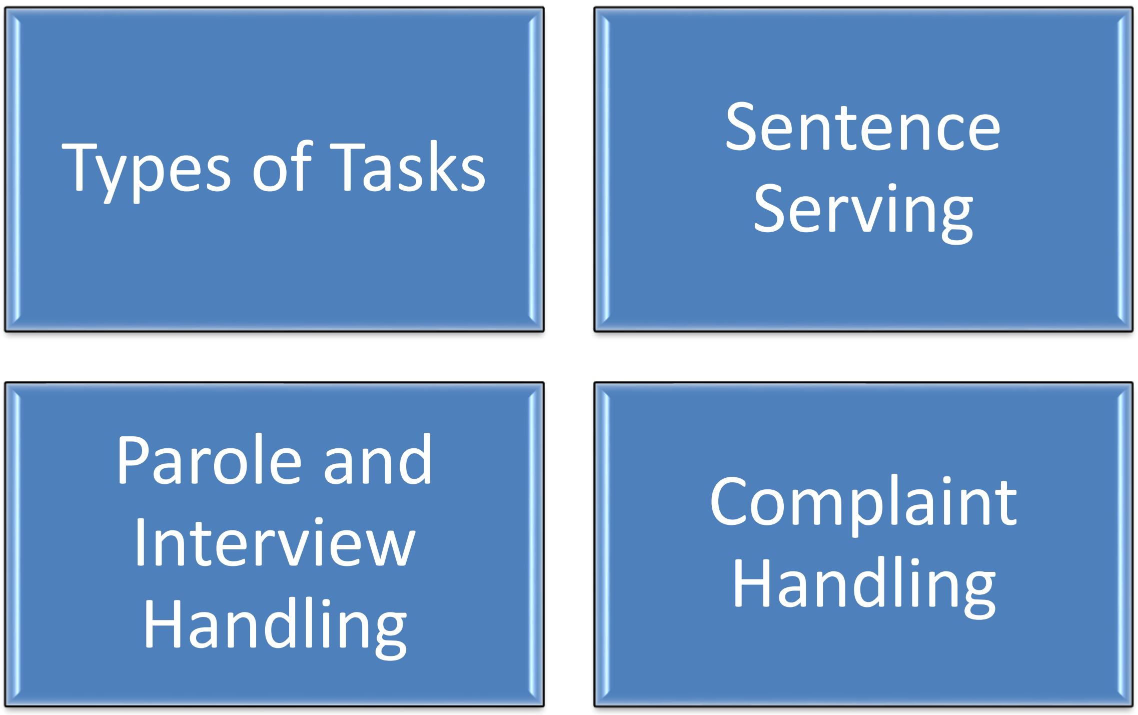


Figure 1

3.2 Types of Tasks

The tasks done by the inmates is mainly divided into two categories-

3.2.1 Mandatory Tasks-

Those tasks which are mandatory for all inmates under the policies of the jail authorities fall under the category of mandatory tasks.
The tasks are to be assigned on rotational basis to serve the purpose of the program.

The following tasks fall under the category of mandatory tasks-

- Cleaning Tasks
- Serving in mess
- Serving in canteen
- Attending meditation sessions
- Taking up psychiatrist sessions
- Attending feedback sessions

This sub-module is implemented as follows-

- Rotation of tasks will be performed on weekly basis.
- Database will save all the information on which task an inmate has performed each week for effective rotational assignment of task.

3.2.2 Specialization Tasks-

- An inmate can pursue either an educational degree/course or be a part of industry/profession such as painting, bakery, hand-made products, apparel, textile, furniture, etc.
- Recruitment Drive also takes place for inmates about to complete their sentence. (Except for rape convicts) Various companies such as Tajmahal Group, Aariz Media, Vedanta Foundation, People's Own Foundation, etc. have offered salaries ranging from Rs. 8,000 to Rs. 35,000 per month.

This sub-module is implemented as follows-

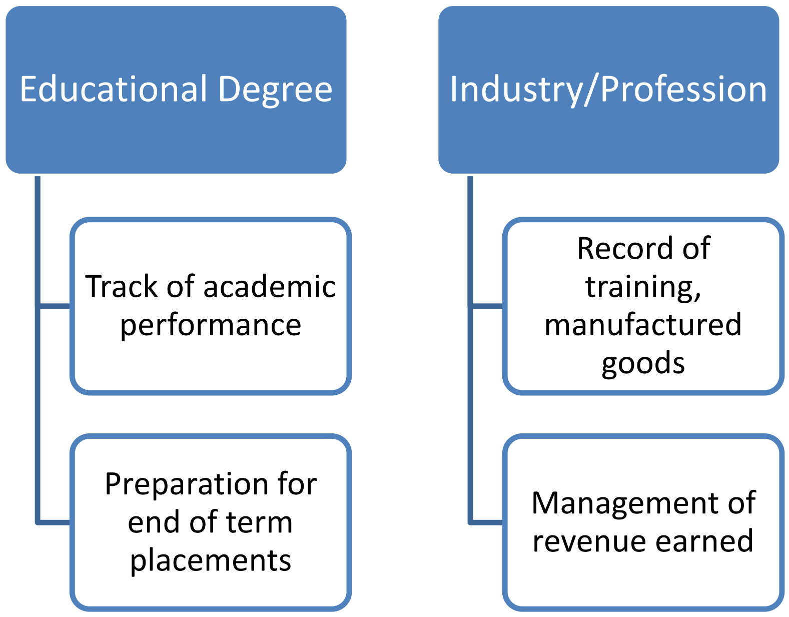

Educational Degree
Track of academic performance
Preparation for end of term placements
Industry/Profession
Record of training, manufactured goods
Management of revenue earned

Figure 2

3.3 Sentence Serving

- Two Important aspects-

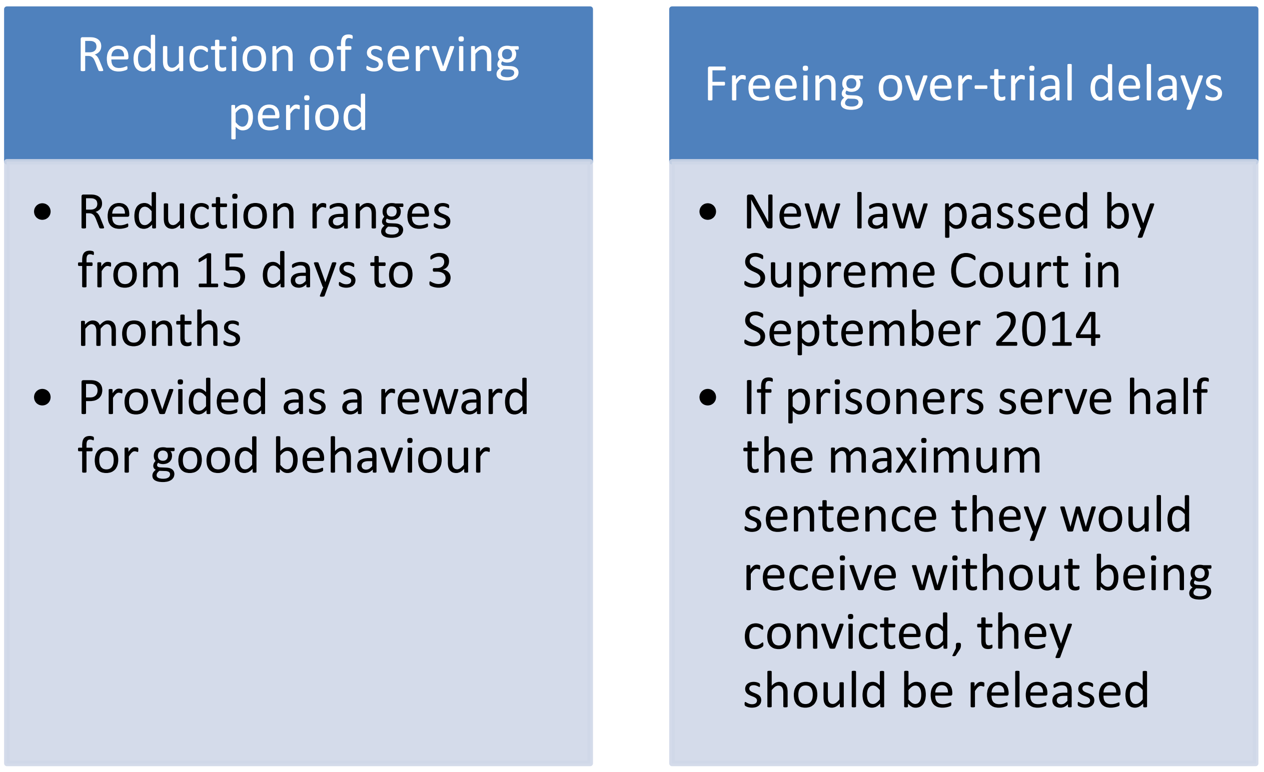

Reduction of serving period
Reduction ranges from 15 days to 3 months
Provided as a reward for good behaviour
Freeing over-trial delays
New law passed by Supreme Court in September 2014
If prisoners serve half the maximum sentence they would receive without being convicted, they should be released

Figure 3

- Reduction of serving period is due to good behaviour which may constitute the following-

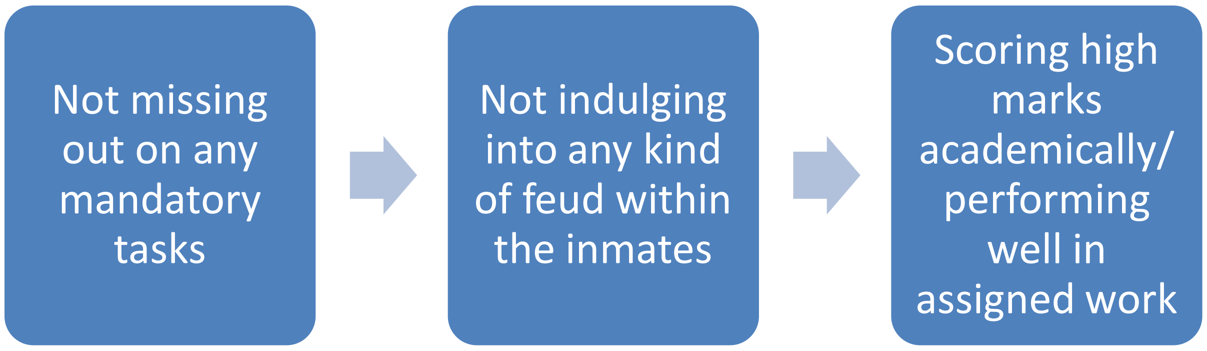

Not missing out on any mandatory tasks
Not indulging into any kind of feud within the inmates
Scoring high marks academically/ performing well in assigned work

Figure 4

- This sub-module is implemented as follows-
    - Database has attributes with such information, so that evaluation of such prisoners is a much easier tasks and much more fair. This information is also crucial during the recruitment drive.
    - Freeing prisoners who serve more than half of the maximum sentence they could have received without being convicted, is done to
        1. Reduce the problem of overcrowding
        2. Release prisoners who serve long sentences for petty crimes, just because they are awaiting trial
    - Calculation of maximum sentence and sentence served taking in account whether the prisoner has been convicted or not will make the procedure swifter.

3.4 Parole and Interview Handling

- An interview is a process where a prisoner can meet his/her family within the jail premises or outside (in case the prisoner is admitted in a hospital). A prisoner is allowed two interviews per week.
- Problem in such a procedure arises in large capacity jails, such as Tihar Jail itself, since a limited space has to be utilized for interview purposes.
- Parole is a system that allows temporary release of prisoners before the completion of their sentence for reasons such as marriage in family, ill health, property disputes, etc.
- Problem arises due to lack of alert system and record management, where paroles are offered with unfair priorities and not monitored properly (as in case of Manu Sharma- convict of Jessica Lall murder case).
- This sub-module is implemented as follows-

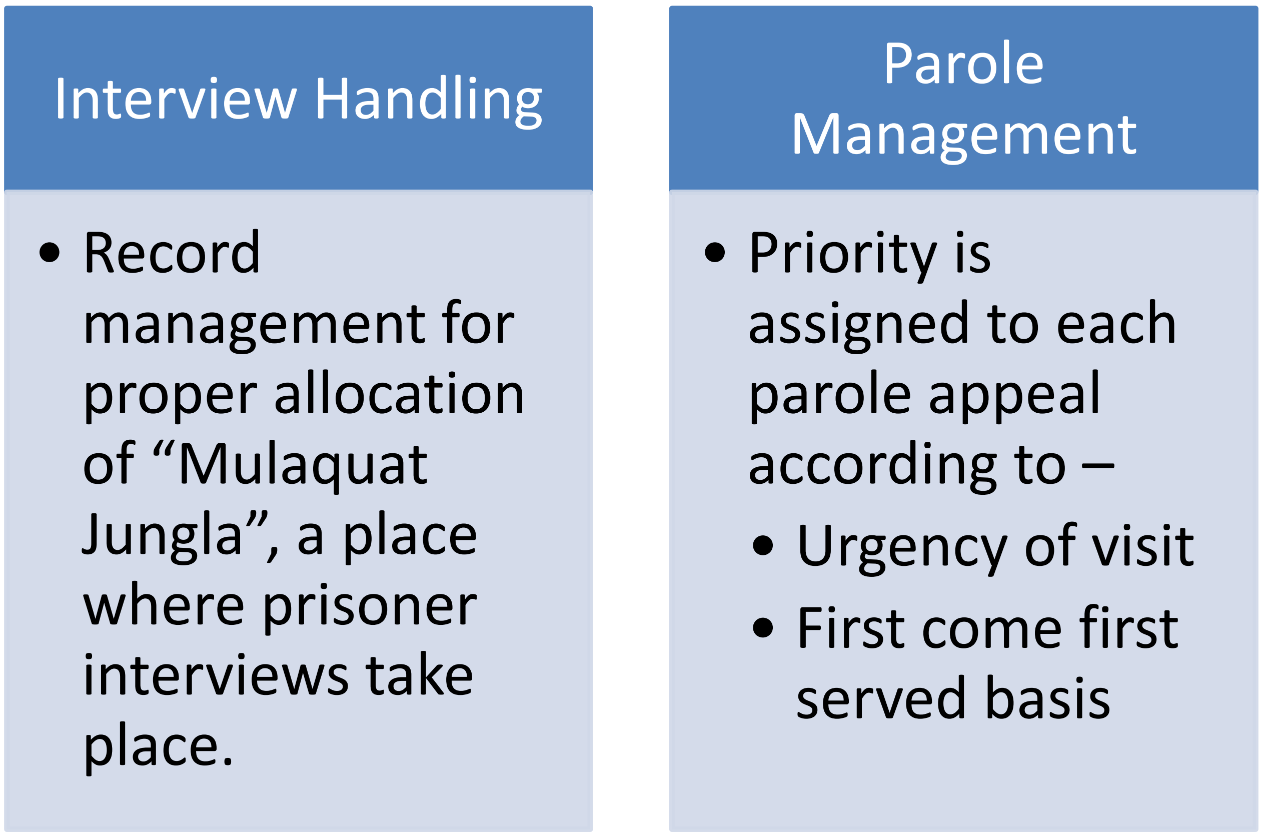

Interview Handling
• Record management for proper allocation of “Mulaquat Jungla”, a place where prisoner interviews take place.
Parole Management
• Priority is assigned to each parole appeal according to –
• Urgency of visit
• First come first served basis

Figure 5

3.5 Complaint Handling

- Traditionally, prisoners drop their grievances or complaints for lacking in the system in a complaint box.
- A box of hand-written complaints is difficult to sort and is a time-consuming process.
- Another method is a group meeting where all the grievances are heard and solutions provided.
- Again, in such cases, lack of permanent record of grievances leads to ignorance of numerous problems.
- This sub-module is implemented as following-
  A system where prisoner enters/ or a police in-charge enters the grievances of a prisoner and also enters the priority (urgent/not so serious) and also calculates the similar number of complaints so that right priority can be assigned to the complaints.

Chapter 4- Implementation

4.1 Web Framework Used- Django

4.1.1 Introduction to Django Web Framework

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so we can focus on writing our app without needing to reinvent the wheel. It's free and open source. written in Python, which follows the model–view–controller architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven web application. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include Pinterest, Instagram, Mozilla, The Washington Times, Disqus, and the Public Broadcasting Service.

4.1.2 History

Django was born in the fall of 2003, when the web programmers at the *Lawrence Journal-World* newspaper, Adrian Holovaty and Simon Willison, began using Python to build applications. It was released publicly under a BSD license in July 2005. The framework was named after guitarist Django Reinhardt.

In June 2008, it was announced that a newly formed Django Software Foundation (DSF) would maintain Django in the future.

4.1.3 Features

4.1.3.1 Components

The core Django MVC framework[4][5] consists of an object-relational mapper which mediates between data models (defined as Python classes) and a relational database ("**M**odel"); a system for processing requests with a web templating system ("**V**iew") and a regular-expression-based URL dispatcher ("**C**ontroller").

Also included in the core framework are:

- A lightweight and standalone web server for development and testing.
- A form serialization and validation system which can translate between HTML forms and values suitable for storage in the database.
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming.
- A caching framework which can use any of several cache methods.
- Support for middleware classes which can intervene at various stages of request processing and carry out custom functions.
- An internal dispatcher system which allows components of an application to communicate events to each other via pre-defined signals.
- An internationalization system, including translations of Django's own components into a variety of languages.
- A serialization system which can produce and read XML and/or JSON representations of Django model instances.

- A system for extending the capabilities of the template engine.
- An interface to Python's built in unit test framework.

4.1.3.2 Bundled applications

The main Django distribution also bundles a number of applications in its "contrib" package, including:

- An extensible authentication system.
- The dynamic administrative interface.
- Tools for generating RSS and Atom syndication feeds.
- A flexible commenting system.
- A sites framework that allows one Django installation to run multiple websites, each with their own content and applications.
- Tools for generating Google Sitemaps.
- Built-in mitigation for cross-site request forgery, cross-site scripting, SQL injection, password cracking and other typical web attacks, most of them turned on by default.[14][15]
- A framework for creating GIS applications.

4.1.3.3 Server Arrangements

Django can be run in conjunction with Apache, NGINX using WSGI, Gunicorn, or Cherokee using flup (a Python module). Django also includes the ability to launch aFastCGI server, enabling use behind any web server which supports FastCGI, such as Lighttpd or Hiawatha. It is also possible to use other WSGI-compliant web servers. Django officially supports four database backends: PostgreSQL, MySQL, SQLite, and Oracle. Microsoft SQL Server can be used with django-mssql but only in Microsoftoperating systems, while similarly external backends exist for IBM DB2, SQL Anywhere and Firebird. There is a fork named django-nonrel which supports NoSQLdatabases, such as MongoDB and Google App Engine's Datastore.

Django may also be run in conjunction with Jython on any Java EE application server such as GlassFish or JBoss. In this case django-jython must be installed in order to provide JDBC drivers for database connectivity, which also provides functionality to compile Django in to a .war suitable for deployment.Google App Engine includes support for Django version 1.x.x as one of the bundled frameworks.

4.1.4 Layers in Django

4.1.4.1 The Model Layer

Django provides an abstraction layer (the “models”) for structuring and manipulating the data of your Web application. Learn more about it below:

- **Models:** Model syntax | Field types | Meta options
- **QuerySets:** Executing queries | QuerySet method reference | Query-related classes | Lookup expressions

- **Model instances:** Instance methods | Accessing related objects
- **Migrations:** Introduction to Migrations | Operations reference | SchemaEditor
- **Advanced:** Managers | Raw SQL | Transactions | Aggregation | Custom fields | Multiple databases | Custom lookups
- **Other:** Supported databases | Legacy databases | Providing initial data | Optimize database access

4.1.4.2 The View Layer

Django has the concept of "views" to encapsulate the logic responsible for processing a user's request and for returning the response. Find all you need to know about views via the links below:

- **The basics:** URLconfs | View functions | Shortcuts | Decorators
- **Reference:** Built-in Views | Request/response objects | TemplateResponse objects
- **File uploads:** Overview | File objects | Storage API | Managing files | Custom storage
- **Class-based views:** Overview | Built-in display views | Built-in editing views | Using mixins | API reference | Flattened index
- **Advanced:** Generating CSV | Generating PDF
- **Middleware:** Overview | Built-in middleware classes

4.1.4.3 The Template Layer

The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. Learn how this syntax can be used by designers and how it can be extended by programmers:

- **For designers:** Syntax overview | Built-in tags and filters | Web design helpers | Humanization
- **For programmers:** Template API | Custom tags and filters

4.1.4.4 Forms

Django provides a rich framework to facilitate the creation of forms and the manipulation of form data.

- **The basics:** Overview | Form API | Built-in fields | Built-in widgets
- **Advanced:** Forms for models | Integrating media | Formsets | Customizing validation
- **Extras:** Form preview | Form wizard

4.1.5 Development Process

Learn about the various components and tools to help you in the development and testing of Django applications:

- Settings: Overview | Full list of settings
- Applications: Overview

- Exceptions: Overview
- django-admin.py and manage.py: Overview | Adding custom commands
- Testing: Introduction | Writing and running tests | Included testing tools | Advanced topics
- Deployment: Overview | WSGI servers | FastCGI/SCGI/AJP (deprecated) | Deploying static files | Tracking code errors by email

4.1.6 The Admin

Django's most popular features:

4.1.6.1 Admin Site: One of the most powerful parts of Django is the automatic admin interface. It reads metadata in your model to provide a powerful and production-ready interface that content producers can immediately use to start adding content to the site. In this document, we discuss how to activate, use and customize Django's admin interface.

Overview

- The admin is enabled in the default project template used by startproject.
- Changed in Django 1.6:
- In previous versions, the admin wasn't enabled by default.
- For reference, here are the requirements:
- Add 'django.contrib.admin' to your INSTALLED_APPS setting.
- The admin has four dependencies - django.contrib.auth, django.contrib.contenttypes,django.contrib.messages and django.contrib.sessions. If these applications are not in yourINSTALLED_APPS list, add them.
- Add django.contrib.messages.context_processors.messages to TEMPLATE_CONTEXT_PROCESSORS as well as django.contrib.auth.middleware.AuthenticationMiddleware anddjango.contrib.messages.middleware.MessageMiddleware to MIDDLEWARE_CLASSES. (These are all active by default, so you only need to do this if you've manually tweaked the settings.)
- Determine which of your application's models should be editable in the admin interface.
- For each of those models, optionally create a ModelAdmin class that encapsulates the customized admin functionality and options for that particular model.
- Instantiate an AdminSite and tell it about each of your models and ModelAdmin classes.
- Hook the AdminSite instance into your URLconf.
- After you've taken these steps, you'll be able to use your Django admin site by visiting the URL you hooked it into (/admin/, by default).
- Admin actions
- Admin documentation generator

4.1.6.2 Admin Actions

The basic workflow of Django’s admin is, in a nutshell, “select an object, then change it.” This works well for a majority of use cases. However, if you need to make the same change to many objects at once, this workflow can be quite tedious.

In these cases, Django’s admin lets you write and register “actions” – simple functions that get called with a list of objects selected on the change list page.

If you look at any change list in the admin, you’ll see this feature in action; Django ships with a “delete selected objects” action available to all models. For example, here’s the user module from Django’s built-in django.contrib.auth app:

4.1.6.3 The Django Admin Documentation Generator

Django’s admindocs app pulls documentation from the docstrings of models, views, template tags, and template filters for any app in INSTALLED_APPS and makes that documentation available from the Django admin.

You may, to some extent, utilize admindocs to quickly document your own code. This has limited usage, however, as the app is primarily intended for documenting templates, template tags, and filters. For example, model methods that require arguments are purposefully omitted from the documentation because they can’t be invoked from templates. The app can still be useful since it doesn’t require you to write any extra documentation (besides docstrings) and is conveniently available from the Django admin.

Overview

To activate the admindocs, you will need to do the following:

- Add django.contrib.admindocs to your INSTALLED_APPS.
- Add (r'^admin/doc/', include('django.contrib.admindocs.urls')) to your urlpatterns. Make sure it’s included *before* the r'^admin/' entry, so that requests to /admin/doc/ don’t get handled by the latter entry.
- Install the docutils Python module (http://docutils.sf.net/).
- Optional: Using the admindocs bookmarklets requiresdjango.contrib.admindocs.middleware.XViewMiddleware to be installed.

Once those steps are complete, you can start browsing the documentation by going to your admin interface and clicking the “Documentation” link in the upper right of the page.

4.2 Database Used- SQLite

4.2.1 Introduction to SQLite

SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others. SQLite has bindings to many programming languages.

4.2.2 History

D. Richard Hipp designed SQLite in the spring of 2000 while working for General Dynamics on contract with the United States Navy. Hipp was designing software used aboard guided missile destroyers, which were originally based on HP-UX with an IBM Informix database back-end. The design goals of SQLite were to allow the program to be operated without installing a database management system or requiring a database administrator. Hipp based the syntax and semantics on PostgreSQL 6.5 documentation. In August 2000, version 1.0 of SQLite was released, with storage based on gdbm (GNU Database Manager). SQLite 2.0 replaced gdbm with a custom B-tree implementation, adding transaction capability. SQLite 3.0, partially funded by America Online, added internationalization, manifest typing, and other major improvements. In 2011 Hipp announced his plans to add an UnQL interface to SQLite databases and to develop *UnQLite*, an embeddable document-oriented database.

4.2.3 Features

- Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Full SQL implementation with advanced features like partial indexes and common table expressions. (Omitted features)
- A complete database is stored in a single cross-platform disk file. Great for use as an application file format.
- Supports terabyte-sized databases and gigabyte-sized strings and blobs. (See limits.html.)
- Small code footprint: less than 500KiB fully configured or much less with optional features omitted.
- Simple, easy to use API.
- Written in ANSI-C. TCL bindings included. Bindings for dozens of other languages available separately.
- Well-commented source code with 100% branch test coverage.

- Available as a single ANSI-C source-code file that is easy to compile and hence is easy to add into a larger project.
- Self-contained: no external dependencies.
- Cross-platform: Android, *BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, WinRT) are supported out of the box. Easy to port to other systems.
- Sources are in the public domain. Use for any purpose.
- Comes with a standalone command-line interface (CLI) client that can be used to administer SQLite databases.

4.2.4 Suggested Use for SQLite:

- **Embedded devices and the internet of things**

Because an SQLite database requires no administration, it works well in devices that must operate without expert human support. SQLite is a good fit for use in cellphones, set-top boxes, televisions, game consoles, cameras, watches, kitchen appliances, thermostats, automobiles, machine tools, airplanes, remote sensors, drones, medical devices, and robots: the "internet of things".

Client/server database engines are designed to live inside a lovingly-attended datacenter at the core of the network. SQLite works there too, but SQLite also thrives at the edge of the network, fending for itself while providing fast and reliable data services to applications that would otherwise have dodgy connectivity.

- **Application file format**

SQLite is often used as the on-disk file format for desktop applications such as version control systems, financial analysis tools, media cataloging and editing suites, CAD packages, record keeping programs, and so forth. The traditional File/Open operation calls sqlite3_open() to attach to the database file. Updates happen automatically as application content is revised so the File/Save menu option becomes superfluous. The File/Save_As menu option can be implemented using the backup API.

There are many benefits to this approach, including improved application performance, reduced cost and complexity, and improved reliability. See technical notes here and here for details.

- **Websites**

SQLite works great as the database engine for most low to medium traffic websites (which is to say, most websites). The amount of web traffic that SQLite can handle depends on how heavily the website uses its database. Generally speaking, any site that gets fewer than 100K hits/day should work fine with SQLite. The 100K hits/day figure is a conservative estimate, not a hard upper bound. SQLite has been demonstrated to work with 10 times that amount of traffic.

The SQLite website (https://www.sqlite.org/) uses SQLite itself, of course, and as of this writing (2015) it handles about 400K to 500K HTTP requests per day, about 15-20% of which are dynamic pages touching the database. Each dynamic page does roughly 200 SQL statements. This setup runs on a single VM that shares a physical server with 23 others and yet still keeps the load average below 0.1 most of the time.

- **Data analysis**

People who understand SQL can employ the sqlite3 command-line shell (or various third-party SQLite access programs) to analyze large datasets. Raw data can be imported from CSV files, then that data can be sliced and diced to generate a myriad of summary reports. More complex analysis can be done using simple scripts written in Tcl or Python (both of which come with SQLite built-in) or in R or other languages using readily available adaptors. Possible uses include website log analysis, sports statistics analysis, compilation of programming metrics, and analysis of experimental results. Many bioinformatics researchers use SQLite in this way.

The same thing can be done with an enterprise client/server database, of course. The advantage of SQLite is that it is easier to install and use and the resulting database is a single file that can be written to a USB memory stick or emailed to a colleague.

- **Cache for enterprise data**

Many applications use SQLite as a cache of relevant content from an enterprise RDBMS. This reduces latency, since most queries now occur against the local cache and avoid a network round-trip. It also reduces the load on the network and on the central database server. And in many cases, it means that the client-side application can continue operating during network outages.

- **Server-side database**

Systems designers report success using SQLite as a data store on server applications running in the datacenter, or in other words, using SQLite as the underlying storage engine for an application-specific database server.

With this pattern, the overall system is still client/server: clients send requests to the server and get back replies over the network. But instead of sending generic SQL and getting back raw table content, the client requests and server responses are high-level and application-specific. The server translates requests into multiple SQL queries, gathers the results, does post-processing, filtering, and analysis, then constructs a high-level reply containing only the essential information.

Developers report that SQLite is often faster than a client/server SQL database engine in this scenario. Database requests are serialized by the server, so concurrency is not an issue. Concurrency is also improved by "database sharding": using separate database files for different subdomains. For example, the server might have a separate SQLite database for each user, so that the server can handle hundreds or thousands of simultaneous connections, but each SQLite database is only used by one connection.

- **File archives**

The SQLite Archiver project shows how SQLite can be used as a substitute for ZIP archives or Tarballs. An archive of files stored in SQLite is only very slightly larger, and in some cases actually smaller, than the equivalent ZIP archive. And an SQLite archive features incremental and atomic updating and the ability to store much richer metadata.

SQLite archives are useful as the distribution format for software or content updates that are broadcast to many clients. Variations on this idea are used, for example, to transmit TV programming guides to set-top boxes and to send over-the-air updates to vehicle navigation systems.

- **Replacement for *ad hoc* disk files**

Many programs use fopen(), fread(), and fwrite() to create and manage files of data in home-grown formats. SQLite works particularly well as a replacement for these *ad hoc* data files.

- **Internal or temporary databases**

For programs that have a lot of data that must be sifted and sorted in diverse ways, it is often easier and quicker to load the data into an in-memory SQLite database and use queries with joins and ORDER BY clauses to extract the data in the form and order needed rather than to try to code the same operations manually. Using an SQL database internally in this way also gives the program greater flexibility since new columns and indices can be added without having to recode every query.

- **Stand-in for an enterprise database during demos or testing**

Client applications typically use a generic database interface that allows connections to various SQL database engines. It makes good sense to include SQLite in the mix of supported databases and to statically link the SQLite engine in with the client. That way the client program can be used standalone with an SQLite data file for testing or for demonstrations.

- **Education and Training**

Because it is simple to setup and use (installation is trivial: just copy the **sqlite3** or **sqlite3.exe** executable to the target machine and run it) SQLite makes a good database engine for use in teaching SQL. Students can easily create as many databases as they like and can email databases to the instructor for comments or grading. For more advanced students who are interested in studying how an RDBMS is implemented, the modular and well-commented and documented SQLite code can serve as a good basis.

- **Experimental SQL language extensions**

The simple, modular design of SQLite makes it a good platform for prototyping new, experimental database language features or ideas.

4.3 Webserver Used- Django Development Server

4.3.1 Introduction

The Django development server (also called the “runserver” after the command that launches it) is a built-in, lightweight Web server you can use while developing your site. It’s included with Django so you can develop your site rapidly, without having to deal with configuring your production server (e.g., Apache) until you’re ready for production. The development server watches your code and automatically reloads it, making it easy for you to change your code without needing to restart anything.

The Django development server comes packaged with Django and can be run with the following command:

```
django-admin.py runserver
```

This launches a lightweight web server running by default using localhost on port 8000. You can modify this by passing something like the following in as an added parameter:

```
django-admin.py runserver 10.0.0.150:8001
```

This will make it so that the web server is serving up your Django application using the IP 10.0.0.150 on port 8001. While there are various other options you can pass in, the above is the most common.

The Django development server is just that, a development server. It was not meant for a production environment, as stated in the official Django documentation here. What it is meant for, however, is to be used as an easy way to test your application in a development environment without the added overhead of installing/configuring a full on web server.

The Django development server shines in that it is light weight and dead simple to use. It serves up your static files without needing to collect them to any specific location as well. The development server also restarts each time you save your files. This is helpful because most all web servers for Python cache your app to get rid of the need for startup times, but require a restart if the code changes so a new copy can be cached.

The obvious downside to the Django development server is that it is not production ready and should only be used for development purposes. It is not made to handle lots of requests or load at any given time. Other than this, it is a great option for those seeking a fast way to get up and running with Django.

Chapter 5- Benefits

5.1 Benefits of the Prison Management System

- This system has immense support for the prison administration. The system is facilitated to provide and receive factually correct, accurate, and timely data and information and without time consuming and errors prone records and registers of prison related activities. It provides an easy-to-use and timely delivery of information. The system provides assistance in task assignment and rotation, thus replacing the tedious and erroneous manual process. It also removes redundant data entry and management and facilitating natural data flow without unnecessary repetition. It has resulted into reduction of paper work, fostering speedy and efficient work processes with accuracy, and error free release of data calculation.
- Overall, it's not just computerization but complete automation and an intelligent system which calculates the sentences, remission, release dates automatically and without human errors. The new system is actively providing support to the transformation of the state prison system towards modern administration. It helps in taking right decisions at the right time. It has fostered the process of more transparency within the system through the above described services.
- The implementation of new rule by the Supreme Court has made this system the most updated among all similar systems. Also it addresses another problem of complaint handling which has not been dealt with yet. The module "Complaint Handling" makes the feedback process more effective and efficient and not just a mere formality.
- The module "Parole and Interview Handling" is also a new feature which will utilize "Mulaquat Jungla" efficiently and provide each prisoner an equal chance to meet their relatives/friends and also reduce the administration work of allocating the premises, and reduce the chance of error. Also the parole handling will check current unfair issue of paroles and delayed appeals do not take place.

5.2 Benefits of Using Django Web Framework

- **Ridiculously fast**

Django was designed to help developers take applications from concept to completion as quickly as possible.

- **Reassuringly secure**

Django takes security seriously and helps developers avoid many common security mistakes.

- **Exceedingly scalable**

Some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale.

5.3 Benefits of Using SQLite

- **Speed**
  - In many cases, 2-3 times faster than MySQL/PostgreSQL
  - Fast PHP Interface
  - No socket and/or TCP/IP overhead
- **Functionality**
  - Sub-selects, Triggers, Transactions, Views.

    - Up to 2TB of data storage.
    - Small memory footprint.
    - Self-contained: no external dependencies.
    - Atomic commit and rollback protect data integrity.
    - Easily movable database.
- **Security**
    - Each user has its own completely independent database

5.4 Benefits of Using Django Development Server

- **Easy to Deploy-** There would be no requirement of installing another web server and hence deployment can take place immediately.
- **Reduction in System Error-** Due to less load on the web framework, system and connection errors would be less
- **A Web Accelerator-** providing SSL and SPDY acceleration, HTTP connection optimization, high-performance caching and HTTP compression.
- **A Load Balancer and Application Delivery solution-** giving reliability, control and consistent performance for HTTP and TCP applications.

Chapter 6-Project Code

Models.py-

```
from django.utils import timezone


class pdetail(models.Model):
        prisoner_name=models.CharField(max_length=50)
        prisoner_add=models.CharField(max_length=500)
        emergency_contact=models.IntegerField()
        gender=models.CharField(max_length=1)
        crime=models.CharField(max_length=50,default=None)
        original_sentence_days=models.IntegerField(default=0)
        potential_sentence_days=models.IntegerField(default=0)
        date_of_conviction=models.DateTimeField(default=None)
        def __str__(self):
                return self.prisoner_name
        def status(self):
                if self.original_sentence_days==0:
                        if self.date_of_conviction <= timezone.now() -
datetime.timedelta(days=self.potential_sentence_days/2):
                                return "release"
                        else:
                                return "under trial"
                else:
                        if self.date_of_conviction<=timezone.now() -
datetime.timedelta(days=self.original_sentence_days):
                                return "release"
                return "convicted"
class pcomplaint(models.Model):
        prisoner_name=models.ForeignKey(pdetail)
```

```
    department=models.CharField(max_length=20)
    complaint_text=models.CharField(max_length=200)
    date_of_complaint=models.DateTimeField('date of complaint')
    def type_of_complaint(self):
        if self.department=="dispute" or self.department=="appliance":
            return "personal"
        elif self.department=="food" or self.department=="hygiene" or self.department=="medical
attention":
            return "basic"
        else:
            return "additional"
    def priority_of_complaint(self):
        if self.type_of_complaint() == "basic":
            return 1
        elif self.type_of_complaint() == "personal":
            if self.date_of_complaint <= timezone.now() - datetime.timedelta(days=15):
                return 1
            return 2
        elif self.type_of_complaint() == "additional":
            if self.date_of_complaint <= timezone.now() - datetime.timedelta(days=30):
                return 1
            elif self.date_of_complaint <= timezone.now() - datetime.timedelta(days=15):
                return 2
            return 3
class parole(models.Model):
    prisoner_name=models.ForeignKey(pdetail)
    parole_reason=models.CharField(max_length=20)
    date_of_application=models.DateTimeField()
    date_of_parole_sanction=models.DateTimeField(default = timezone.now() -
datetime.timedelta(days=5400))
```

```
    parole_days=models.IntegerField(default=0)

    prisoner_return=models.CharField(max_length=3, default="NA")

    reason_text=models.CharField(max_length=200)

    def parole_priority(self):

        if self.parole_reason=="death in family":

            return 1

        elif self.parole_reason=="family event":

            return 2

        else:

            return 3

    def parole_status(self):

        if self.date_of_parole_sanction<=timezone.now() - datetime.timedelta(days=3650):

            return "to be sanctioned"

        elif self.date_of_parole_sanction>=timezone.now() -
datetime.timedelta(days=self.parole_days):

            return "parole ongoing"

        elif self.date_of_parole_sanction<=timezone.now() -
datetime.timedelta(days=self.parole_days) and self.prisoner_return=="Yes":

            return "parole over"

        else:

            return "call back inmate!"

class task(models.Model):

    prisoner_name=models.ForeignKey(pdetail)

    regular_task=("mess cleaning task","mess serving task","mess dishes task","canteen cleaning task",
"canteen serving task", "canteen dishes task", "washroom cleaning task", "appliance cleaning task", "corridor
cleaning task")

    degree=models.CharField(max_length=30, default="NA")

    profession=models.CharField(max_length=30, default="NA")

    def specialization_task_status(self):

        if self.degree=="NA" and self.degree=="NA":

            if pdetail.crime=="rape":
```

```
                        return "Task cannot be assigned"
                else:
                        return "Assign Task"
            else:
                    return "Task assigned"
        def regular_task_assignment(self):
          x=random.randomint(1,9)
          return regular_task[x]
```

Admin.py-

```
from django.contrib import admin
from msystem.models import pdetail,pcomplaint,parole,task
class pdetailAdmin(admin.ModelAdmin):
        list_display=('prisoner_name','gender','crime','status')
        list_filter=['date_of_conviction']



class pcomplaintAdmin(admin.ModelAdmin):
        list_display=('prisoner_name','date_of_complaint','type_of_complaint','priority_of_complaint')
        list_filter=['date_of_complaint']


class paroleAdmin(admin.ModelAdmin):
        list_display=('prisoner_name','parole_reason','date_of_application','parole_priority','parole_status')
        list_filter=['date_of_application']


class taskAdmin(admin.ModelAdmin):
        list_display=('prisoner_name','degree','profession', 'specialization_task_status')


admin.site.register(pdetail,pdetailAdmin)
admin.site.register(pcomplaint,pcomplaintAdmin)
admin.site.register(parole,paroleAdmin)

admin.site.register(task,taskAdmin)
```

Settings.py-

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(__file__))

TEMPLATE_DIRS = [os.path.join(BASE_DIR,'templates')]


# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/1.7/howto/deployment/checklist/


# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = 'w+@oru0$p_vns*%%m3nax&)2=1$5w*rcq)4e+1bj#lka&d#0z$'


# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True


TEMPLATE_DEBUG = True


ALLOWED_HOSTS = []



# Application definition


INSTALLED_APPS = (

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',
    'django.contrib.staticfiles',
            'msystem',
)

MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
)

ROOT_URLCONF = 'PMS.urls'

WSGI_APPLICATION = 'PMS.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.7/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Internationalization
# https://docs.djangoproject.com/en/1.7/topics/i18n/


LANGUAGE_CODE = 'en-us'


TIME_ZONE = 'UTC+5:30'


USE_I18N = True


USE_L10N = True


USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.7/howto/static-files/


STATIC_URL = '/static/'
```

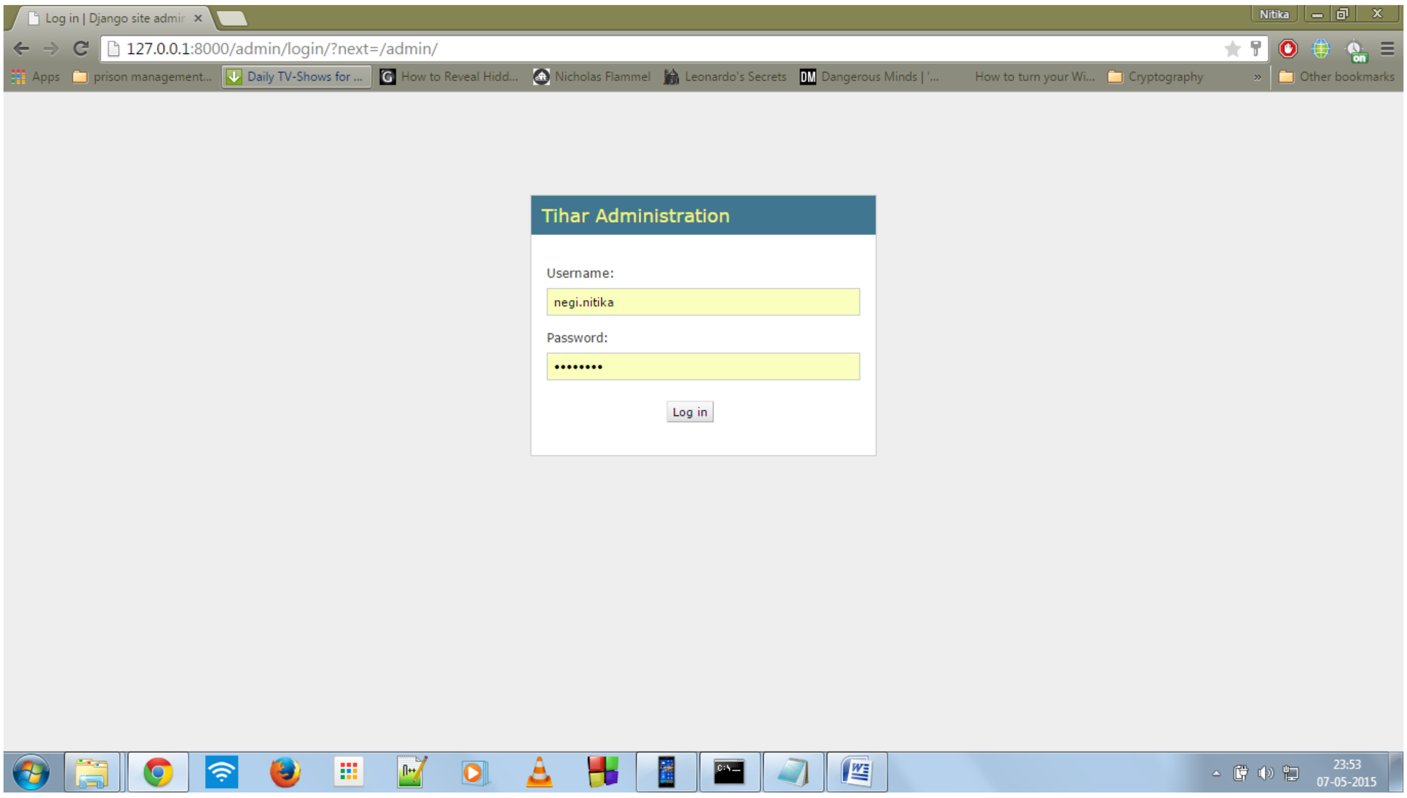
Base_site.html

```
{% extends "admin/base.html" %}


{% block title %}{{ title }} | {{ site_title|default:_('Django site admin') }}{% endblock %}


{% block branding %}
<h1 id="site-name"><a href="{% url 'admin:index' %}">Tihar Administration</a></h1>
{% endblock %}


{% block nav-global %}{% endblock %}
```

Manage.py-

```
#!/usr/bin/env python

import os

import sys



if __name__ == "__main__":

    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "PMS.settings")



    from django.core.management import execute_from_command_line



    execute_from_command_line(sys.argv)
```

Chapter 7-Results



Tihar Administration

Username:

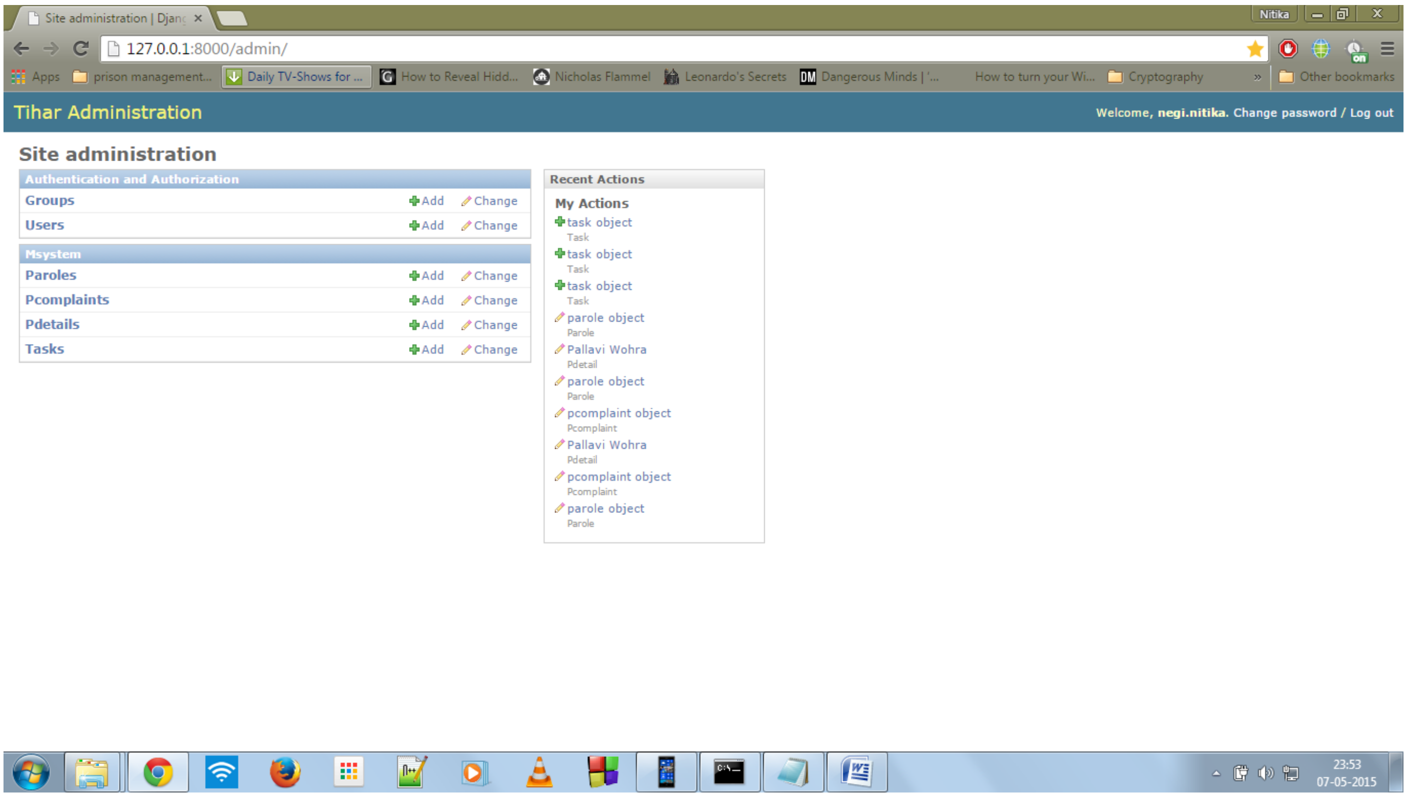
negi.nitika

Password:

••••••••

Log in

Figure 6



Tihar Administration

Welcome, **negi.nitika**. Change password / Log out

Site administration

Authentication and Authorization		
Groups	Add	Change
Users	Add	Change

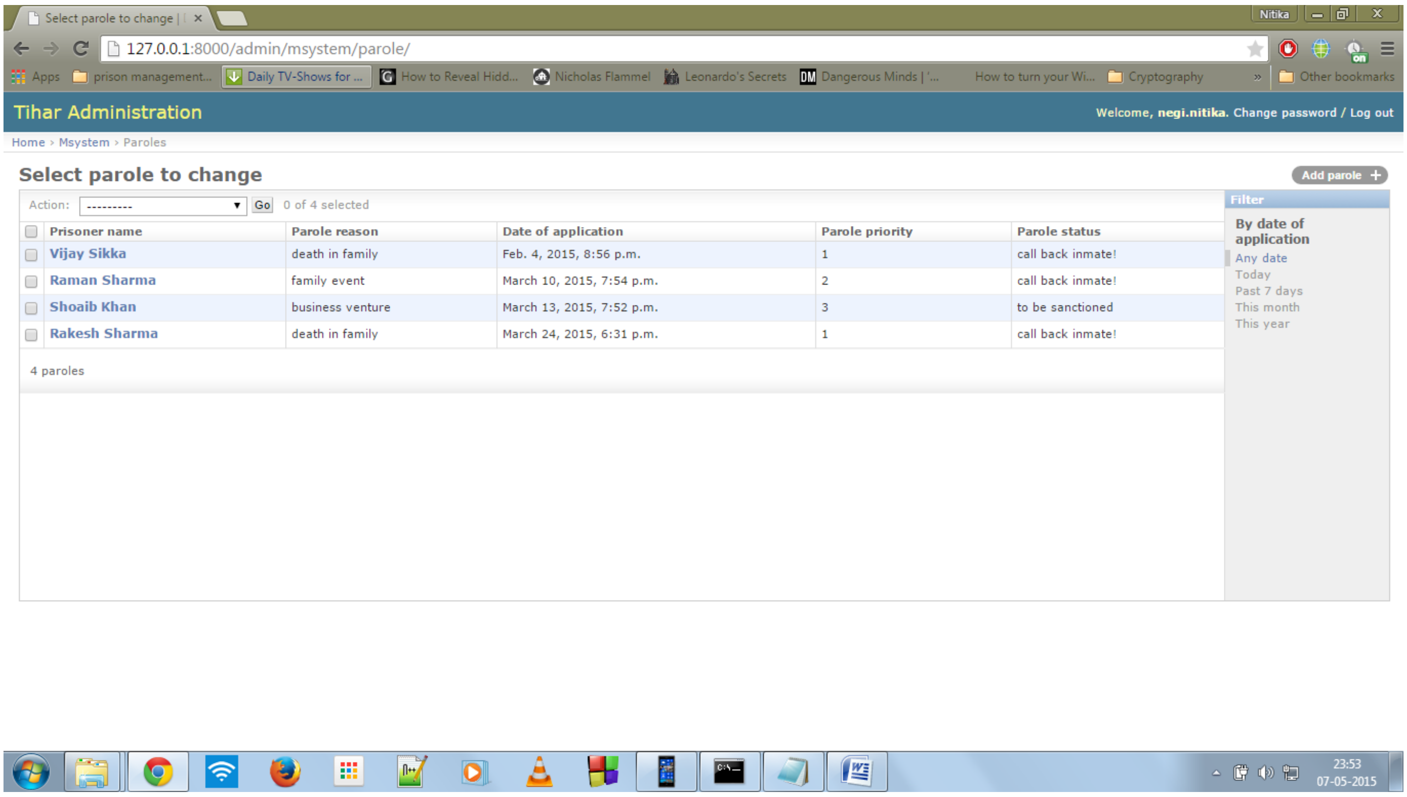
Msystem		
Paroles	Add	Change
Pcomplaints	Add	Change
Pdetails	Add	Change
Tasks	Add	Change

Recent Actions

My Actions

- task object
  Task
- task object
  Task
- task object
  Task
- parole object
  Parole
- Pallavi Wohra
  Pdetail
- parole object
  Parole
- pcomplaint object
  Pcomplaint
- Pallavi Wohra
  Pdetail
- pcomplaint object
  Pcomplaint
- parole object
  Parole

Figure 7



Tihar Administration

Welcome, negi.nitika. Change password / Log out

Home › Msystem › Paroles

Select parole to change

Action: --------- Go 0 of 4 selected

Prisoner name	Parole reason	Date of application	Parole priority	Parole status
Vijay Sikka	death in family	Feb. 4, 2015, 8:56 p.m.	1	call back inmate!
Raman Sharma	family event	March 10, 2015, 7:54 p.m.	2	call back inmate!
Shoaib Khan	business venture	March 13, 2015, 7:52 p.m.	3	to be sanctioned
Rakesh Sharma	death in family	March 24, 2015, 6:31 p.m.	1	call back inmate!

4 paroles

Filter

By date of application

Any date

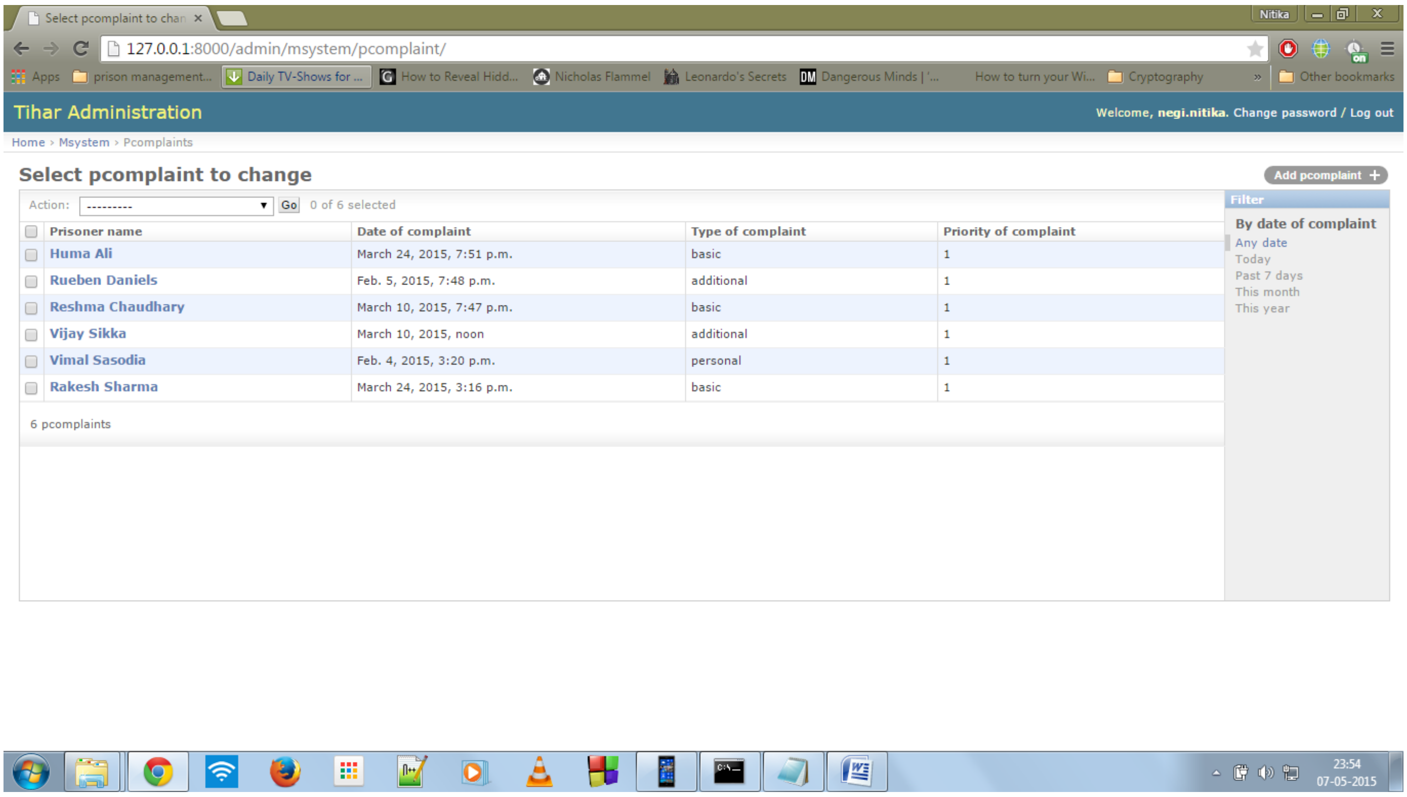
Today

Past 7 days

This month

This year

Figure 8



Tihar Administration

Welcome, negi.nitika. Change password / Log out

Home › Msystem › Pcomplaints

Select pcomplaint to change

Action: --------- Go 0 of 6 selected

Prisoner name	Date of complaint	Type of complaint	Priority of complaint
Huma Ali	March 24, 2015, 7:51 p.m.	basic	1
Rueben Daniels	Feb. 5, 2015, 7:48 p.m.	additional	1
Reshma Chaudhary	March 10, 2015, 7:47 p.m.	basic	1
Vijay Sikka	March 10, 2015, noon	additional	1
Vimal Sasodia	Feb. 4, 2015, 3:20 p.m.	personal	1
Rakesh Sharma	March 24, 2015, 3:16 p.m.	basic	1

6 pcomplaints

Filter

By date of complaint

Any date

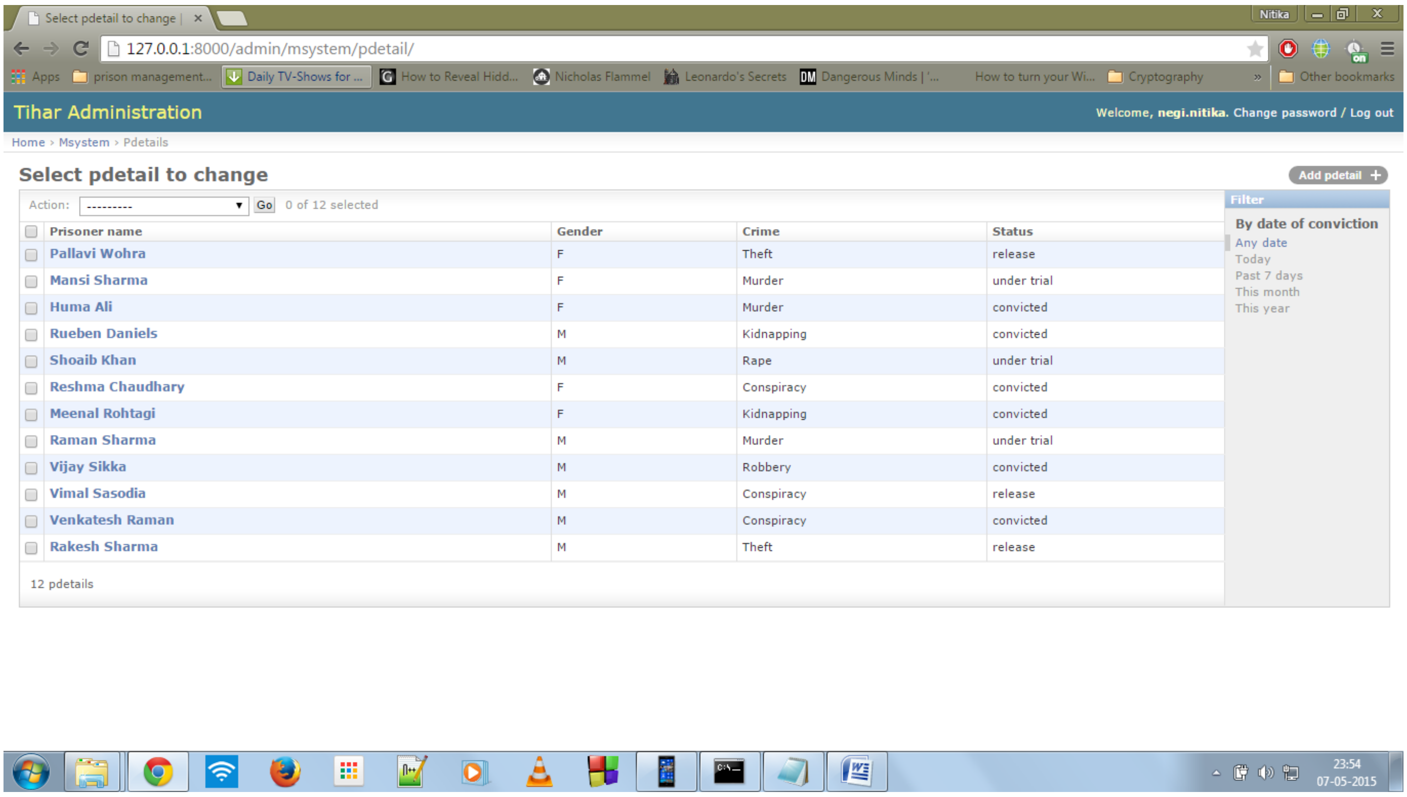
Today

Past 7 days

This month

This year

Figure 9



Tihar Administration

Welcome, negi.nitika. Change password / Log out

Home › Msystem › Pdetails

Select pdetail to change

Action: --------- Go 0 of 12 selected

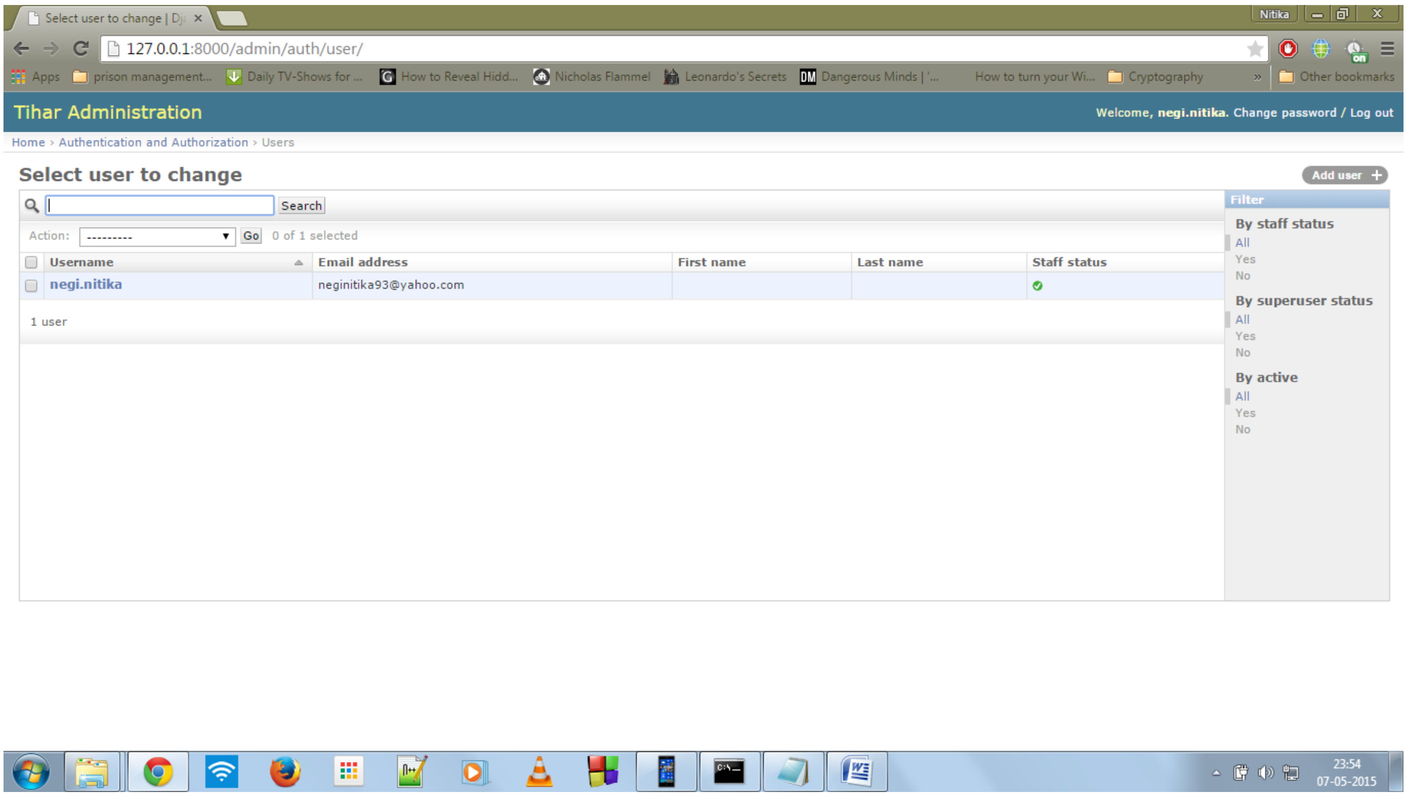
Prisoner name	Gender	Crime	Status
Pallavi Wohra	F	Theft	release
Mansi Sharma	F	Murder	under trial
Huma Ali	F	Murder	convicted
Rueben Daniels	M	Kidnapping	convicted
Shoaib Khan	M	Rape	under trial
Reshma Chaudhary	F	Conspiracy	convicted
Meenal Rohtagi	F	Kidnapping	convicted
Raman Sharma	M	Murder	under trial
Vijay Sikka	M	Robbery	convicted
Vimal Sasodia	M	Conspiracy	release
Venkatesh Raman	M	Conspiracy	convicted
Rakesh Sharma	M	Theft	release

12 pdetails

Filter

By date of conviction
- Any date
- Today
- Past 7 days
- This month
- This year

Figure 10



Tihar Administration

Welcome, negi.nitika. Change password / Log out

Home › Authentication and Authorization › Users

Select user to change

Search

Action: --------- Go 0 of 1 selected

Username	Email address	First name	Last name	Staff status
negi.nitika	neginitika93@yahoo.com			✓

1 user

Filter

By staff status
- All
- Yes
- No

By superuser status
- All
- Yes
- No

By active
- All
- Yes
- No

Figure 11

Chapter 8- Scalability

Prison Management System is a scalable project which can be deployed to various other prison systems across the country to streamline the process of managing inmates within the prison premises. It can also be modified whenever the Supreme Court releases any new laws for holding of prisoners and hence can be used for a very long time without being behind technological advances. Introduction of new module will also be a relatively easier task and hence, this project can be scaled to the extent where the prison and its activities be controlled completely via computers. This vision goes hand-in-hand with the vision of our Prime Minister Shri Narendra Modi, i.e.- “Digital India”.

Conclusion

In this project report I highlighted how Prison Management works across the country and beyond. I also mentioned how Tihar Jail, by the effort of Kiran Bedi, has completely changed its course from being a place of punishment to a place of reformations and hope. This project will contribute to that vision and also is a step towards “Digital India”. The four modules focus on basic and unattended matters of prison management and also ensure minimization of corruption. Prison Management System is highly scalable and has a high scope for future expansion.

References

List of Papers/Case Studies/Articles-

1. ***Paper 1- Prison Models and Prison Management Models and the Texas Prison System*** submitted by Gevana Lynn Salinas (Texas State University- San Marcos, Dept. Of Political Science, Public Administration Program), Summer 2009
2. ***Paper 2- Prison Management System for Arthur Road Jail*** in Mumbai, Maharashtra, India. Submitted by Rahul Singh, Sr. Consultant, KPMG, 2010
3. ***Case Study- Prison Management System (PRISMS) An e-Governance Project of the Govt. of Goa*** By Osama Manzar 2011-2012
4. ***Article***- Tihar Jail by Kiran Bedi (Adapted by Ananya Parthibhan) taken from her autobiography "I DARE!"

Other References-

1. delhi.govt.in
2. wikipedia.org/wiki/Tihar_Jail
3. tihartj.nic.in
4. The wall street journal
5. ndtv.com/articles/india
6. https://www.djangoproject.com/
7. http://www.sqlite.org/
8. ipi.org.in/texts/nsip/nsip-full/tihar
9. dailymail.co.uk/indiahome/indianews
10. https://www.djangoproject.com/
11. thehindu.com
12. timesofindia.indiatimes.com