POCKET CERTIFICATE

A Project Report

Submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF COMPUTER APPLICATION

Submitted by

Group Members:

MOHAMMED SHENES H.K (Reg. No: 193102)
ARAFAN C.P (Reg. No: 193133)
FAHEEM ABDURAHIMAN K.P (Reg. No: 193134)
MOHAMMED NASEEF BIN MASHOOD (Reg. No: 193145)
MOHAMMED SALAH SAINUDHEEN (Reg. No: 193147)
MUHAMMED ANAN A.P (Reg. No: 193148)
MUHAMMED NIZAMUDEEN A.P (Reg. No: 193150)

DEPARTMENT OF COMPUTER SCIENCE

JAMIA HAMDARD KANNUR CAMPUS

JUNE 2021

CERTIFICATE

This is to certify that the Project Report entitled **"POCKET CERTIFICATE"** submitted to the Department of Computer Science, Jamia Hamdard Kannur Campus in partial fulfilment of the requirement for the award of Bachelor of Computer Application is an original work carried out by members of Group 1 in the year 2021 and that the major project has not been submitted to any University/Institute for the fulfilment of the requirement of any course of study.

Signature of the project guide

Name: Mrs. Prajina B

Designation: Lecturer at JHK

Place: Kannur

DECLARATION

I hereby declare that the report of the project work entitled **"POCKET CERTIFICATE"** submitted to the Department of Computer Science, Jamia Hamdard Kannur Campus in partial fulfilment of the requirement for the award of Bachelor of Computer Application is a bonafide work carried out by the members of Group 1. The material contained in this report has not been submitted to any University/Institute for the fulfilment of the requirements of any course of study.

Group 1 members

MOHAMMED SHENES H.K

ARAFAN C.P

FAHEEM ABDURAHIMAN K.P

MOHAMMED NASEEF BIN MASHOOD

MOHAMMED SALAH SAINUDHEEN

MUHAMMED ANAN A.P

MUHAMMED NIZAMUDEEN A.P

ACKNOWLEDGEMENT

During the period that I worked on this project, I relied on the assistance, resources and support of many people. I acknowledge with gratitude the help, guidance and encouragement given by many individualsin the completion of this work.

I register my gratitude to our respected principal, **Dr TP Mammootty,** for all the help and guidance.

I express my sincere thanks to **Mrs Devapriya E D**, the Course Coordinator of the Department of Computer Science.

I express my sincere thanks to **Mrs Prajina B,** Lecturer in the Department of Computer Science, and project guide, for her expert guidance, constant motivation and valuable pieces of advice during the development of this project.

I am thankful to **Mrs Zeena Anuya V M K**, Assistant Professor of the Department of Computer Science, for the valuable suggestions during the completion of this project.

I am also thankful to **Mrs Reena K P**, the Assistant Professor of the Department of Computer Science, for her kind cooperation and help throughout the work.

I am also thankful to the staff of other Departments for every help and cooperation in the venture. I am very thankful to my parents and dear friends for their moral support.

Above all, I thank the Lord Almighty who gave good health, courage and interest to complete the work successfully.

MOHAMMED SHENES H.K
ARAFAN C.P
FAHEEM ABDURAHIMAN K.P
MOHAMMED NASEEF BIN MASHOOD
MOHAMMED SALAH SAINUDHEEN
MUHAMMED ANAN A.P
MUHAMMED NIZAMUDEEN A.P

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Introduction

Almost all Indian government-issued documents are in physical form across the country. Be it Certificates like Caste Certificate, Income Certificate, Marriage Certificate and so on. This means every time a resident need to share the document with an agency to avail of any service, an attested photocopy or an original physical copy is shared. This is a laborious task when it comes to the safekeeping and organising of these documents. With the rise in natural disasters as we've seen in recent years, the securement of these important documents has been quite difficult to maintain. A lot of people have had great difficulties with documents that were washed away in disasters like the recent floods. Even besides that, these documents don't always stand time with damages that inevitably occur over a period.

The use of document hard copies also creates huge overhead in terms of manual verification, paper storage, manual audits, etc. incurring high cost and inconvenience for the offices that have to work with them.

There has also been the issue of the pandemic that requires us to stay socially distant at all times as much as possible. In times like these, going from village office to office, coming across people from all wakes of the place waiting for their requirements among them can be dangerous.

This is where our system comes into play.

The **Pocket Certificate** System is a software, which tries to alter the originality of the Government-Related Documents such as Nativity, Community, Income and Marriage Certificates into encrypted form. Whenever the user wants to download a file, the system will decrypt the document which is stored on the server. It also allows the user to apply for these documents to the corresponding offices and access them as required from the safety and security of their own home.

2. SYSTEM ANALYSIS

Systems analysis is "the process of studying a procedure or business in order to identify its goals and purposes and create system and procedures that will achieve them in an efficient way". Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the purpose of studying how well those components work and interact to accomplish their purpose.

The field of system analysis relates closely to requirements analysis or operation research. It is also "an explicit formal inquiry carried out to help a decision-maker identify a better course of action and make a better decision than they might otherwise have made."

The terms analysis and synthesis stem from Greek, meaning "to take apart" and "to put together," respectively. These terms are used in many scientific disciplines, from mathematics and logic to economics and psychology, to denote similar investigative procedures. The analysis is defined as "the procedure by whichwe break down an intellectual or substantial whole into parts," while synthesis means "the procedure by which we combine separate elements or components to form a coherent whole." System analysis researchers apply the methodology to the systems involved, forming an overall picture.

System analysis is used in every field where something is developed. Analysis can also be a series of components that perform organic functions together, such as system engineering. System engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed.

The development of a computer-based information system includes a system analysis phase. This helps produce the data model, a precursor to creating or enhancing a database. There are a number of different approaches to system analysis. When a computer-based information system is developed, system analysis (according to the Waterfall model) would constitute the following steps:

- The development of a feasibility study: determining whether a project is economically, socially, technologically and organizationally feasible
- Fact-finding measures, designed to ascertain the requirements of the system's end-users (typically involving interviews, questionnaires, or visual observations of work on the existing system)
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

Another view outlines a phased approach to the process. This approach breaks system analysis into 5 phases:

- Scope Definition: Clearly defined objectives and requirements necessary to meet a project's requirements as defined by its stakeholders.
- Problem analysis: the process of understanding problems and needs and arriving at solutions that meet them.
- Requirements analysis: determining the conditions that need to be met.
- Logical design: looking at the logical relationship among the objects.
- Decision analysis: making a final decision.

Use cases are widely used system analysis modelling tools for identifying and expressing the functional requirements of a system. Each use case is a business scenario or event for which the system must provide a defined response. Use cases evolved from the object-oriented analysis.

2.1 Existing system:

The existing system for attaining Government-issued documents is not a reliable one. It involves the resident needing to use physical forms of documents or proofs either as photocopies or original copies. This is inconvenient and time-consuming. The residents are required to waste hours of their working day waiting in busy Government offices. In the current times of Covid-19, this is a risk. Safekeeping of the documents in their physical form is also an issue with the increasing risk of natural disasters like floods.

2.2 Proposed system:

To overcome the disadvantages of the existing application, we propose a new certificate management system: **Pocket Certificate.**

This proposed system is more reliable and efficient. It offers the users a secure and trustworthy system. It is less time-consuming in the processing phase, and easier to get documents verified by Government officials. The documents are also much safer with the existence of original digital copies in secure servers that can be downloaded any time.

Goals of the proposed system:

i. The system should be easy to operate.
ii. The working in the organization will be well planned and organized.
iii. The level of accuracy in the proposed system will be higher.
iv. The reliability of the proposed system will be high due to proper storage of Information.
v. Provide quick and efficient retrieval of information.

Advantages:

1. Less time consuming
2. Cost-effective
3. High accuracy
4. High efficiency
5. Avoids redundancy
6. User friendly
7. Maintains security
8. Enables to view a large amount of data in a short time

Disadvantages:

1. High starting cost requires.
2. Additional manpower is necessary.
3. Data communication system will have an additional cost

2.3 Feasibility study:

A feasibility study is an assessment of the practicality of a proposed project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

A well-designed feasibility study should provide a historical background of the business or project, a description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation. A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of thestudy for potential investors and lending institutions. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

A project feasibility study is a comprehensive report that examines in detail the five frames of analysis of a given project. It also takes into consideration its four Ps, its risks and POVs, and its constraints (calendar, costs, and norms of quality). The goal is to determine whether the project should go ahead, be redesigned, or else be abandoned altogether. The five frames of analysis are the frame of definition; the frame of contextual risks; the frame of potentiality; the parametric frame; the frame of dominant and contingency strategies.

The four Ps are traditionally defined as Plan, Processes, People, and Power. The risks are considered to be external to the project (e.g., weather conditions) and are divided into eight categories: (Plan) financial and organizational (e.g., government structure for a private project); (Processes) environmental and technological; (People) marketing and sociocultural; and (Power) legal and political. POVs are Points of Vulnerability: they differ from risks in the sense that they are internal to the project and can be controlled or else eliminated.

The constraints are the standard constraints of calendar, costs and norms of quality that can each be objectively determined and measured along the entire project lifecycle. Depending on projects, portions of the study may suffice to produce a feasibility study; smaller projects, for example, may not require an exhaustive environmental assessment.

TYPES OF FEASIBILITY STUDY:

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are fivetypes of feasibility study—separate areas that a feasibility study examines, described below.

1. Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building—currently, this project is not technically feasible.

2. Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3. Legal Feasibility

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal locationisn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

4. Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether—and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine howa project plan satisfies the requirements identified in the requirements analysis phase of system development.

5. Scheduling Feasibility

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

Importance of Feasibility Study:

The importance of a feasibility study is based on organizational desire to "get it right" before committing resources, time, or budget. A feasibility study might uncover new ideas that could completely change a project's scope. It's best to make these determinations in advance, rather than to jump in and learn that the project won't work. Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of the proposed project.

Benefits of A Feasibility Study:

Below are the benefits of doing a feasibility study in project management:

- Get a clear-cut idea of whether the project is likely to be successful, before allocating budget, manpower, and time.
- Enhances the project teams' efficiency and focus
- Helps detect and capitalize on new opportunities
- Substantiates with evidence of why and how a project should be executed
- Streamlines the business alternatives
- Diagnoses errors and aids in troubleshooting them
- Prevents threats from occurring and helps in risk mitigation
- Gives valuable insights to both the team and stakeholders associated with the project.

Steps to conduct a feasibility study:

The following stages are involved while conducting any feasibility study, in general:

- A preliminary analysis: This is like a pre-screening of the project. It helps discover the viability of the project as well as identify any roadblocks if any.
- Scope definition: This step includes outlining the project's scope as well as its potential impact on the organization.
- Market research: This is an essential factor, as no project is begun without adequate market research. A thorough analysis of the existing market and competition is done to manage the project accordingly.
- Financial assessment: In this stage, all the costs related to the project, including equipment, man-hours, the financial risks, and the benefits associated with the project are estimated and scrutinized.
- Alternative solutions: Whenever any hiccups arise, the team should be well-prepared to come up with a solution. This is an integral yet dynamic part of a feasibility study.

- Go/no-go decision: The final stage of a feasibility study is the course of action, in other words, whether the project is worth proceeding with or not.

3.REQUIREMENT SPECIFICATIONS

A Requirement Specification is a collection of the set of all requirements that are to be imposed on the design and verification of the product. The specification also contains other related information necessaryfor the design, verification, and maintenance of the product.

Requirement specification process:

It's the process of writing down the user and system requirements into a document. The requirements should be clear, easy to understand, complete and consistent. In the first iteration, you specify the user requirements, then you specify more detailed system requirements.

Four major steps of requirements specification:

1) Elicitation: The Elicitation step is where the requirements are first gathered. It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc.

2) Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.

3) Specification: During this step, the analyst prioritizes and formally documents the requirements in a Requirements Definition Report.

4) Validation: Validation involves techniques to confirm that the correct set of requirements has been specified to build a solution that satisfies the project's business objectives.

A software requirements specification (SRS) is a description of a software system to be developed. Also known as a stakeholder requirements specification (StRS). The software requirements specification laysout functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these rolesmay be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to

reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document lists sufficient and necessary requirements for the projectdevelopment. To derive the requirements, the developer needs to have a clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with theproject team and customer throughout the software development process.

The SRS may be one of a contract's deliverable data item descriptions or have other forms of organizationally-mandated content. Typically, an SRS is written by a technical writer, a systems architect, or a software programmer.

A System Requirements Specification (SyRS) (abbreviated SysRS when need to be distinct from a software requirements specification (SRS)) is a structured collection of information that embodies the requirements of the system.

A business analyst (BA), sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, the BA typically performs a liaison function between the business side of anenterprise and the information technology department or external service providers.

Requirements analysis is a very critical process that enables the success of a system or software project to beassessed. Requirements are generally split into two types:

Functional and Non-Functional Requirements

Functional Requirements:

These are the requirements that the end-user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in thefinal product, unlike the non-functional requirements.

Non-functional requirements:

These are the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another. They are also

called non-behavioural requirements. They deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Domain requirements:

Domain requirements are the requirements that are characteristic of a particular category or domain of projects. The basic functions that a system of a specific domain must necessarily exhibit come under this category. For instance, in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user-specific.

3.1 Hardware requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in the case of operating systems. An HCL lists tested compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Minimum Specifications

- Processor : Intel/AMD 6 core 6 thread processor with 9MB cache latency.
- Ram : 4 GB DDR3 2666 MHZ
- HDD : 2 GB available space
- Keyboard : Windows compatible
- Mouse : Windows compatible
- OS : Microsoft® Windows® 7/8/10 (64-bit)
- Display : 1280 x 800 minimum screen resolution.

Recommended Specifications

- Processor : Intel i7 sky lake processors and up
  AMD ryzen3 2400G
- Ram : 8 GB DDR4 2666 MHZ
- HDD : 4 GB available space
- Keyboard : Windows compatible
- Mouse : Windows compatible
- OS : Microsoft® Windows® 7/8/10 (64-bit)
- Display : 1920 x 1080 recommended screen resolution.

The use of hardware acceleration has additional requirements on Windows and Linux:

- Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel64), and Execute Disable (XD) Bit functionality;
- AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V)and Supplemental Streaming SIMD Extensions 3 (SSSE3);
- AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality.

To work with Android 8.1 (API level 27) and higher system images, an attached webcam must have the capability to capture 720p frames.

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines twosets of system requirements: minimum and recommended. With the increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. A second meaning of the term of system requirements, is a generalization of this first definition, giving the requirements to be met in the design of a system or sub-system.

3.2 Software requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. The requirement for a system is the description of what the system should do, the service or services that it provides and the constraints on its operation. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

- A condition or capability needed by a user to solve a problem or achieve an objective.
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- A documented representation of a condition or capability as in 1 or 2.

The activities related to working with software requirements can broadly be broken down into elicitation,analysis, specification, and management.

An operating system is one of the requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of the same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for MicrosoftWindows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (orat all) on Linux distributions using Kernel v2.2 or v2.4.

Windows XP:

Windows XP is an operating system produced by Microsoft as part of the Windows NT family of operating systems. It was the successor to both Windows 2000 for professional users and Windows Me for home users, and it was released to manufacturing on August 24, 2001, with retail sales beginning on October 25, 2001. It was Microsoft's operating system for use on personal computers such as home and business desktops, laptops, tablet PCs and media centre PCs until the OS having replaced by Windows Vista in February 2007.

Windows 98:

Windows 98 is an operating system developed by Microsoft as part of its Windows 9x family of Microsoft Windows operating systems. It is the successor to Windows 95, and was released to manufacturing on May15, 1998, and generally to retail on June 25, 1998. Like its predecessor, it is a hybrid 16-bit and 32-bit monolithic product with the boot stage based on MS-DOS.

Linux kernel:

The Linux kernel is a free and open-source, monolithic, modular, multitasking, Unix-like operating system kernel. It was conceived and created in 1991 by Linus Torvalds for his i386-based PC, and it was soon adopted as the kernel for the GNU operating system, which was created as a free replacement for UNIX. Since then, it has spawned a plethora of operating system distributions, commonly also called Linux.

Windows 10:

Windows 10 is a series of operating systems developed by Microsoft and released as part of its Windows NT family of operating systems. It is the successor to Windows 8.1, released nearly two years earlier, and was released to manufacturing on July 15, 2015, and broadly released for the general public on July 29, 2015. Windows 10 was made available for download via MSDN and Technet, as a free upgrade for retail copies of Windows 8 and Windows 8.1 users via the Windows Store, and to Windows 7 users via Windows Update. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users, in addition to additional test builds of Windows 10, which are available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

In our project, we have used Windows 10 as our operating system. this project also runs in windows 7/8.

Software specification

- Operating system : Windows 7 or above
- Technology : PYTHON, HTML, JAVA, CSS
- Front End : PYTHON, XML, HTML, JAVA
- Back End : Mysql
- Platform used : JetBrains PyCharm 2017.12.
- Browser : Google chrome, opera, Mozilla
- Editor : Notepad

3.3 Technology Specifications Front End-

Python:

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. Python is often described as a "batteries included" language due to its comprehensive standard library. Python consistently ranks as one of the most popular programming languages. Guido van Rossum began working on Python in the late 1980s, as a successorto the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection systemusing reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backwards-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.

Python was conceived in the late 1980sby Guido van Rossum at Centrum Wiskunde&Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project. Guido van Rossumhas since then withdrawn his nomination for the 2020 Steering Council.

Java:

Java is a high-level, class-based, object-oriented programming language that is designed to have as fewimplementation dependencies as possible. It is a general-purpose programming language intended to let application developers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++ but has fewer low-level facilities than either of them. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform.

The original and reference implementation of Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Oracle offers its own Hotspot Java Virtual Machine; however, the official reference implementation is the OpenJDK JVM which is free open-source software and used by most developers andis the default JVM for almost all Linux distributions. James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. Java was originally designed for interactivetelevision, but it was too advanced for the digital cable television industry at the time. The language was initially called *Oak* after an oak tree that stood outside Gosling's office. Later the project went by the name *Green* and was finally renamed *Java*, from Java coffee, a type of coffee from Indonesia. Gosling designed Java with a C/C++style syntax that system and application programmers would find familiar.

Sun Microsystems released the first public implementation as Java 1.0 in 1996 It promised **Write Once,Run Anywhere** (**WORA**) functionality, providing no-cost run-times on popular platforms. Fairly secureand featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular. The Java 1.0 compiler was re-written in Java by Arthur van Hoff to comply strictly with the Java 1.0 language specification. With the advent of Java 2 (released initially as J2SE 1.2 in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. J2EE included technologies and APIs for enterprise applications typically run in server environments, while J2ME featuredAPIs optimized for mobile applications. The desktop version was renamed J2SE. In 2006, for marketing purposes, Sun renamed new J2 versions as *Java EE*, *Java ME*, and *Java SE*, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC 1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process. Java remains a *de facto* standard, controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java throughthe selling of licenses for specialized products such as the Java Enterprise System. On November 13, 2006, Sun released much of its Java virtual machine (JVM) as free and open-source software (FOSS), under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of its JVM's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright. Sun's vice-president Rich Green said that Sun's ideal role with regard to Java was as an *evangelist*. Following Oracle Corporation's acquisition of Sun Microsystems in2009–10, Oracle has described itself as the steward of Java technology with a relentless commitment to fostering a community of participation and transparency. This did not prevent Oracle from filing a lawsuit against Google shortly after that for using Java inside the Android SDK.

HTML:

The **HyperText Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<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 such as JavaScript, which affects the behaviour and content of web pages. The inclusion of CSS defines the look and layout of content. The World WideWeb Consortium (W3C), the former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

CSS:

CSS stands for cascading style sheets. Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongsideHTML and JavaScript.

CSS is the language for describing the presentation of Web pages, including colours, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language.

CSS is not technically programming languages; they're just page structure and style information. But before moving on to JavaScript and other true languages, you need to know the basics of HTML and CSS, as they are on the front end of every web page and application.

XML:

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The World Wide Web Consortium's XML 1.0 Specification of 1998 and several other related specifications. All of the free open

standards define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such asthose used in web services. Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

The essence of why extensible markup languages are necessary is explained in the Markup language (for example, see Markup language XML) and at Standard Generalized Markup Language.

Hundreds of document formats using XML syntax have been developed, including RSS, Atom, SOAP, SVG, and XHTML. XML-based formats have become the defaultfor many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (Open Document), and Apple's iWork.

XML has also provided the base language for communication protocols such as XMPP. Applications for the Microsoft .NET Framework use XML files for configuration, and property lists are an implementation ofconfiguration storage built on XML.

Many industry data standards, such as Health Level 7, OpenTravel Alliance, FpML, MISMO, and National Information Exchange Model are based on XML and the rich features of the XML schema specification. Many of these standards are quite complex and it is not uncommon for a specification to comprise several thousand pages. In publishing, Darwin Information Typing Architecture is an XML industry data standard.XML is used extensively to underpin various publishing formats.

XML is widely used in Service-oriented architecture (SOA). Disparate systems communicate with each other by exchanging XML messages. The message exchange format is standardised as an XML schema (XSD). This is also referred to as the canonical schema. XML has come into common use for the interchange of data over the Internet. IETF RFC:3023, now superseded by RFC:7303, gave rules for the construction of Internet Media Types for use when sending XML. It also defines the media types application/XML and text/XML, which say only that the data is in XML and nothing about its semantics.

RFC 7303 also recommends that XML-based languages be given media types ending in +XML; for example, image/SVG+XML for SVG. Further guidelines for the use of XML in a networked context appear in RFC 3470, also known as IETF BCP 70, a document covering many aspects of designing and deploying an XML-based language.

The XML specification defines an XML document as a well-formed text, meaning that it satisfies a list of syntax rules provided in the specification. Some key points in the fairly long list include:

- The document contains only properly encoded legal Unicode characters.
- None of the special syntax characters such as < and & appear except when performing their markup-delineation roles.
- The start-tag, end-tag, and empty-element tag that delimit elements are correctly nested, with none missing and none overlapping.
- Tag names are case-sensitive; the start-tag and end-tag must match exactly.
- Tag names cannot contain any of the characters !"#$%&'()*+,/;<=>?@[\]^` {|}~, nor a space character, and cannot begin with "-", ".", or a numeric digit.
- A single root element contains all the other elements.

The definition of an XML document excludes texts that contain violations of well-formed rules; they are simply not XML. An XML processor that encounters such a violation is required to report such errors and tocease normal processing. This policy, occasionally referred to as "draconian error handling," stands in notable contrast to the behaviour of programs that process HTML, which is designed to produce a reasonable result even in the presence of severe markup errors. XML's policy in this area has been criticized as a violation of Postel's law ("Be conservative in what you send; be liberal in what you accept").

The XML specification defines a valid XML document as a well-formed XML document that also conforms to the rules of a Document Type Definition (DTD).

In addition to being well-formed, an XML document may be valid. This means that it contains a reference toa Document Type Definition (DTD) and that its elements and attributes are declared in that DTD and followthe grammatical rules for them that the DTD specifies.

XML processors are classified as validating or non-validating depending on whether or not they check XML documents for validity. A processor that discovers a validity error must be able to report it but may continue normal processing.

A DTD is an example of a schema or grammar. Since the initial publication of XML 1.0, there has been substantial work in the area of schema languages for XML. Such schema languages typically constrain the set of elements that may be used in a document, which attributes may be applied to them, the order in whichthey may appear, and the allowable parent/child relationships.

Document type definition:

The oldest schema language for XML is the document type definition (DTD), inherited from SGML. DTDs have the following benefits:

- DTD support is ubiquitous due to its inclusion in the XML 1.0 standard.
- DTDs are terse compared to element-based schema languages and consequently present more information on a single screen.

- DTDs allow the declaration of standard public entity sets for publishing characters.
- DTDs define a document type rather than the types used by a namespace, thus grouping all constraints for a document in a single collection.

DTDs have the following limitations:

- They have no explicit support for newer features of XML, most importantly namespaces.
- They lack expressiveness. XML DTDs are simpler than SGML DTDs and there are certain structures that cannot be expressed with regular grammar. DTDs only support rudimentary datatypes.
- They lack readability. DTD designers typically make heavy use of parameter entities (which behave essentially as textual macros), which make it easier to define complex grammars, but at the expense of clarity.
- They use a syntax based on regular expression syntax, inherited from SGML, to describe the schema. Typical XML APIs such as SAX does not attempt to offer applications a structured representation of the syntax, so it is less accessible to programmers than an element-based syntax may be.

Two peculiar features that distinguish DTDs from other schema types are the syntactic support for embedding a DTD within XML documents and for defining entities, which are arbitrary fragments of text ormarkup that the XML processor inserts in the DTD itself and the XML document wherever they are referenced as character escapes.

DTD technology is still used in many applications because of its ubiquity.

A newer schema language, described by the W3C as the successor of DTDs, is XML Schema, often referredto by the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system and allow for more detailed constraints on an XML document's logical structure. XSDs also use an XML-based format, which makes it possible to use ordinary XML tools to help process them.

Back End

MySQL:

- **MYSQL**

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

MySQL is free and open-source software under the terms of the GNU General Public License and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project tocreate MariaDB.

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL,but more often MySQL is used with other programs to implement applications that need relational database capability. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is also used by many popular websites, including Facebook, Flickr, MediaWiki, Twitter, and YouTube.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, ArcaOS, eComStation, IBM I, IRIX, Linux, macOS, Microsoft Windows, NetBSD, NovellNetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, or a proprietary license.

Support can be obtained from the official manual. Free support additionally is available in different IRCchannels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and price. Additionally, a number of third-party organisations exist to provide support and services.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the averagecase" and that the "developer interfaces are there, and the documentation (not to mention feedback in thereal world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded SQL database server".

History:

MySQL was created by a Swedish company, MySQL AB, founded by Swedes David Axmark, Allan Larsson and Finland Swede Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (proprietarily licensed) mSQL antecedent.

MySQL is offered under two different editions: the open-source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and isbuilt from the same code base.

Major features as available in MySQL 5.6:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors
- Updatable views
- Online Data Definition Language (DDL) when using the InnoDB Storage Engine.
- Information schema
- Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.
- A set of SQL Mode options to control runtime behaviour, including a strict mode to better adhere to SQL standards.
- X/Open XA distributed transaction processing (DTP) support; two-phase commit as part of this, using the default InnoDB storage engine
- Transactions with save points when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.

- ACID compliance when using InnoDB and NDB Cluster Storage Engines
- SSL support
- Query caching
- Sub-SELECTs (i.e., nested SELECTs)

Built-in replication support

- Asynchronous replication: master-slave from one master to many slaves or many masters to one slave
- Semi synchronous replication: Master to slave replication where the master waits on replication.
- Synchronous replication: multi-master replication is provided in MySQL Cluster.
- Virtual Synchronous: Self-managed groups of MySQL servers with multi-master support can be done using: Galera Cluster or the built-in Group Replication plugin
- Full-text indexing and searching
- Embedded database library
- Unicode support
- Partitioned tables with the pruning of partitions in the optimizer
- Shared-nothing clustering through MySQL Cluster
- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
- Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster.
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.
- The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or MySQL's GitHub repository, both under theGPL license.

Platforms:

PyCharm:

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. The app allows you to install and run PyCharm on your android device. IMPORTANT, this app is not PyCharm, but it allows you to INSTALL PyCharm in just one command. All you have to do is download termux, insert command from my app, and install vnc viewer. After executing the code in termux, you canrun PyCharm with a single command. PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

PyCharm is an integrated development environment or IDE. While you don't compile Python code the same way you do Java or C sharp, PyCharm acts like a true IDE for Python, but unlike Visual Studio, PyCharm is geared specifically and only for Python development.

Dreamweaver:

Adobe Dreamweaver is a proprietary web development tool from Adobe Inc. It was created by Macromediain 1997 and developed by them until Macromedia was acquired by Adobe Systems in 2005. Adobe Dreamweaver is available for the macOS and Windows operating systems. It is written in C++. Adobe Dreamweaver is available for the macOS and Windows operating systems.

WampServer:

WampServer refers to a solution stack for the Microsoft Windows operating system, created by Romain Bourdon and consisting of the Apache web server, OpenSSL for SSL support, MySQL database and PHP programming language. Stands for "Windows, Apache, MySQL, and PHP." WAMP is a variation of LAMP for Windows systems and is often installed as a software bundle (Apache, MySQL, and PHP). It is often used for web development and internal testing, but may also be used to serve live websites.

WampServer is a collection of web development tools that you can use to install an Apache server with PHP and MySQL databases.

4. SYSTEM DESIGN

The system design provides an understanding of the procedural details necessary for implementing the system recommended in the feasibility study. Basically, it is all about the creation of a new system. This is a criticalphase since it decides the quality of the system and has a major impact on the testing and implementation phases. System design is the process of designing the elements of a system such as the architecture, modulesand components, the different interfaces of those components and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elementsto enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

Elements of a System:

- **Architecture** - This is the conceptual model that defines the structure, behaviour and more views of a system. We can use flowcharts to represent and illustrate the architecture.
- **Modules** - These are components that handle one specific task in a system. A combination of the modules makes up the system.
- **Components** - This provides a particular function or group of related functions. They are made up of modules.
- **Interfaces** - This is the shared boundary across which the components of the system exchange information and relate.
- **Data** - This is the management of the information and data flow.

Major Tasks Performed During the System Design Process

1. Initialize design definition

- Plan for and identify the technologies that will compose and implement the systems elements and their physical interfaces.
- Determine which technologies and system elements have a risk to become obsolete or evolving during the operation stage of the system. Plan for their potential replacement.

- Document the design definition strategy, including the need for and requirements of any enabling systems, products, or services to perform the design.

2. Establish design characteristics

- Define the design characteristics relating to the architectural characteristics and check that they are implementable.
- Define the interfaces that were not defined by the System Architecture process or need to be defined as the design details evolve.
- Define and document the design characteristics of each system element.

3. Assess alternatives for obtaining system elements

- Assess the design options
- Select the most appropriate alternatives.
- If the decision is made to develop the system element, the rest of the design definition process and the implementation process are used. If the decision is to buy or reuse a system element, the acquisition process may be used to obtain the system element.

4. Manage the design

- Capture and maintain the rationale for all selections among alternatives and decisions for the design,architecture characteristics.
- Assess and control the evolution of the design characteristics.

Factors that Affect Technology Trade-offs during System Design

Scale of Product:

- For example, enterprise software companies that are building system-level software prioritize reliability because customers need to use them. Each change needs to be rigorously tested, and often approved before it can be released.
- Meanwhile, consumer internet companies spend time and money on making their UX delightful so that people want to use them. Reliability is something they're willing to sacrifice. Since many are web-based applications, they can iterate quickly and release changes frequently.

Time:

- Learning new technologies sometimes often takes time. The trade-offs in this instance will be madeaccording to which stack/technology will be in time with the set delivery dates. If switching to a new stack/technology will result in a major shift on the delivery dates and major inconveniences to the stakeholders, then the switch can be held off until an appropriate time.

Cost:

- On a larger scale, Technology decisions are made based on which is more cost-effective, where a comparison can be done on which will be more effective between buying an off the shelf system and customizing it or building a new system.

Efficiency:

- Technology trade-offs are also done based on which technology is more efficient for example choosing between ReactJs or AngularJs for a front-end application.

User Experience and Support:

- The amount of support and documentation available on a given technology can also be a determining factor in the decisions. Working with Technologies that have a large support base, comprehensive documentation and a good user experience is much easier and take a very short time to ramp up on due to a large number of resources available to support it.

Maintainability:

- Maintainability in this case is the ease with which a product can be maintained in order to correct errors, fix bugs and add additional features. Trade-off decisions will be made based on the maintainability of the Technology.

Reliability

- In this case the tradeoffs are made based on the Technology that performs consistently well and consistently upgrades to more efficient versions.

Scalability

- Technology tradeoffs are also made based on the technologies that are more scalable and able to handle increased loads efficiently without a break in the system efficiency.

The system design phase includes:

- Input design
- Output design
- Database design

Characteristics of Design

- A Design should exhibit a hierarchical organization that makes intelligent use of control among components of the software
- A design should be modular that is, the software should be logical
- A design should contain a distinct and separable representation of data and procedure
- A design should lead to an interface that reduces the complexity of the connection between modules and with the external environment.

4.1 Module description

The proposed system has 3 modules

- Admin
- Village Officer
- Clerk

Admin

- Login
- Department management
- Village officer management
- Clerk management
- View feedback
- View complaints and send the reply
  (user and Akshaya)
- Send notification

Village Officer

- Login
- View profile
- View department and clerks
- Approve/Reject application
  (Nativity, Community, Income, etc)
- View application
- View notification

Clerk

- Login
- View certificate category
- View application request
  (Nativity, Community, Income, etc)
- Verify certificate and forward to village officer and view status
- View notification
- View feedback

4.2 Database Design

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data into the database model. Database management system manages the data accordingly.

Database design involves classifying data and identifying interrelationships. This theoretical representation of the data is called an ontology. The ontology is the theory behind the database's design.

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. A properly designed database is easy to maintain, improves data consistency and are cost-effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

The main objectives of database designing are to produce logical and physical designs models of theproposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

It helps produce database systems

1. That meet the requirements of the users
2. Have high performance.

Database designing is crucial to a high-performance database system.

Database Designing

- Logical model - This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
- Physical model - This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

Implementation

- Data conversion and loading - this stage is concerned with importing and converting data from the old system into the new database.
- Testing - this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

Determining data to be stored

In the majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore, the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system.

This process is one which is generally considered part of requirements analysis and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.

Logically structuring data

Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to arrange the data into a logical structure which can then be mapped into the storage objects supported by the database management system. In the case of relational databases, the storageobjects are tables that store data in rows and columns. In an Object database, the storage objects correspond directly to the objects used by the Object-oriented programming language used to write the applications that will manage and access the data. The relationships may be defined as attributes of the object classes involved or as methods that operate on the object classes.

The way this mapping is generally performed is such that each set of related data which depends upon a single object, whether real or abstract, is placed in a table. Relationships between these dependent objects are then stored as links between the various objects.

Each table may represent an implementation of either a logical object or a relationship joining one ormore instances of one or more logical objects. Relationships between tables may then be stored as links connecting child tables with parents. Since complex logical relationships are themselves tables, they will probably have links to more than one parent.

In this project proposed system Pocket Certificate uses database pocket certificate and has the following database tables.

acomplaints:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
acid	int	11		☑	☑	☐	☑	☐		
acomplaint	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
acdate	date			☐	☐	☐	☐	☐		
acuserid	int	11		☐	☐	☐	☐	☐		
areply	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
reply_date	date			☐	☐	☐	☐	☐		
				☐	☐	☐	☐	☐		

akshaya:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
alogin_id	int	11		☐	☐	☐	☐	☐		
aname	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
aemail	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
aphone	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
licensenumber	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
aimage	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

caste_certificate:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
ccid	int	11		☑	☑	☐	☑	☐		
cuserid	int	11		☐	☐	☐	☐	☐		
creligion	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
ccaste	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
category	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
ccfileupload_id	int	11		☐	☐	☐	☐	☐		
status	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
clerkstatus	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

clerk:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
clogin_id	int	11		☐	☐	☐	☐	☐		
c_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_dob	date			☐	☐	☐	☐	☐		
c_hname	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_place	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_pin	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_post	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_district	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_image	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_qualification	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_email	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
cjoining_date	date			☐	☐	☐	☐	☐		
cending_date	date			☐	☐	☐	☐	☐		
c_phone	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
dept_id	int	11		☐	☐	☐	☐	☐		
				☐	☐	☐	☐	☐		

feedback:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
feedback_id	int	11		☑	☑	☐	☑	☐		
fuser_id	int	11		☐	☐	☐	☐	☐		
feedback	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
date	date			☐	☐	☐	☐	☐		
				☐	☐	☐	☐	☐		

department:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
dept_id	int	11		☑	☑	☐	☑	☐		
dept_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
certificate_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
d_details	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
p_id	varchar	100		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

income_certificate:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
inc_id	int	11		☑	☑	☐	☑	☐		
incuserid	int	11		☐	☐	☐	☐	☐		
income_source	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
amount	int	11		☐	☐	☐	☐	☐		
income_fileupload	int	11		☐	☐	☐	☐	☐		
status	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
clerkstatus	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

login:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
login_id	int	11		☑	☑	☐	☑	☐		
username	varchar	200		☑	☑	☐	☐	☐	latin1	latin1_swedish_ci
password	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
usertype	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

ucomplaints:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
ucid	int	11		☑	☑	☐	☑	☐		
complaint	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
c_date	date			☐	☐	☐	☐	☐		
ucuserid	int	11		☐	☐	☐	☐	☐		
reply	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
reply_date	date			☐	☐	☐	☐	☐		
				☐	☐	☐	☐	☐		

marriage_certificate:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
mcid	int	11		☑	☑	☐	☑	☐		
wmcuserid	int	11		☐	☐	☐	☐	☐		
hmcuserid	int	11		☐	☐	☐	☐	☐		
mhousename	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mplace	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mpost	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
m_pin	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mdistrict	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
memail	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mmobile	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mvillage	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mtaluk	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mname_of_local_body	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
marriagedate	date			☐	☐	☐	☐	☐		
mlocation	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
mfileupload	int	11		☐	☐	☐	☐	☐		
status	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
clerkstatus	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

upload_certificate:

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
uid	int	11		☑	☑	☐	☑	☐		
pid	int	11		☐	☐	☐	☐	☐		
userid	int	11		☐	☐	☐	☐	☐		
image	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
				☐	☐	☐	☐	☐		

nativity_certificate:

	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
*	ncid	int	11		☑	☑	☐	☑	☐		
	ncuserid	int	11		☐	☐	☐	☐	☐		
	fcase_state	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	f_district	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	f_taluk	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	f_village	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	m_district	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	m_taluk	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	m_village	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	m_fileupload_id	int	11		☐	☐	☐	☐	☐		
	status	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	clerkstatus	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
					☐	☐	☐	☐	☐		

user:

	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
*	loginid	int	11		☑	☑	☐	☑	☐		
	name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	gender	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	dob	date			☐	☐	☐	☐	☐		
	housename	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	place	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	post	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	pin	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	district	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	email	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	mobile	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	village	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	taluk	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	name_of_local_body	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	father_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	mother_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
					☐	☐	☐	☐	☐		

village_officer:

	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
*	login_id	int	11		☑	☑	☐	☑	☐		
	v_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_dob	date			☐	☐	☐	☐	☐		
	v_hname	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_place	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_post	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_pin	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_district	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_image	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_qualification	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	v_email	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	joining_date	date			☐	☐	☐	☐	☐		
	end_date	date			☐	☐	☐	☐	☐		
	v_phone	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
					☐	☐	☐	☐	☐		

proof:

	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
*	pid	int	11		☑	☑	☐	☑	☐		
	p_name	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
					☐	☐	☐	☐	☐		

notification:

	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Charset	Collation
*	nid	int	11		☑	☑	☐	☑	☐		
	notification	varchar	200		☐	☐	☐	☐	☐	latin1	latin1_swedish_ci
	date	date			☐	☐	☐	☐	☐		
					☐	☐	☐	☐	☐		

4.1 Procedural Design

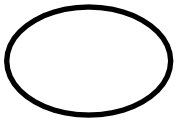
DATA FLOW DIAGRAM: (DFD)

A DFD is a network that describes the flow of data and the processes that change or transform, data throughout a system. The DFD network is a formal, logical abstract of a system that may have many possible physical configurations. A DFD is a structured analysis and design tool that can be used for flowcharting in place of, or in association with, information-oriented and process-oriented system flowcharts.

 : A data flow that connects the output of an object or process to the input of another process

 : It represents the data source or destination.

Or

 : An open-ended rectangle represents a data store that stores data

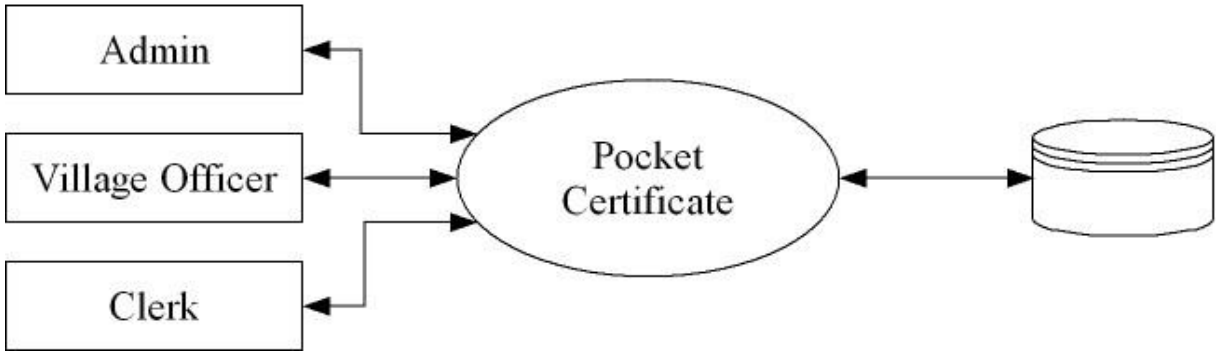
Six rules for constructing the data flow diagram

- Arrows should not cross each other
- Squares, circles, and files must bear a name
- Decomposed data flow squares and circles can have the same names.
- Choose meaningful names for dataflow
- Draw all data flow around the outside of the diagram

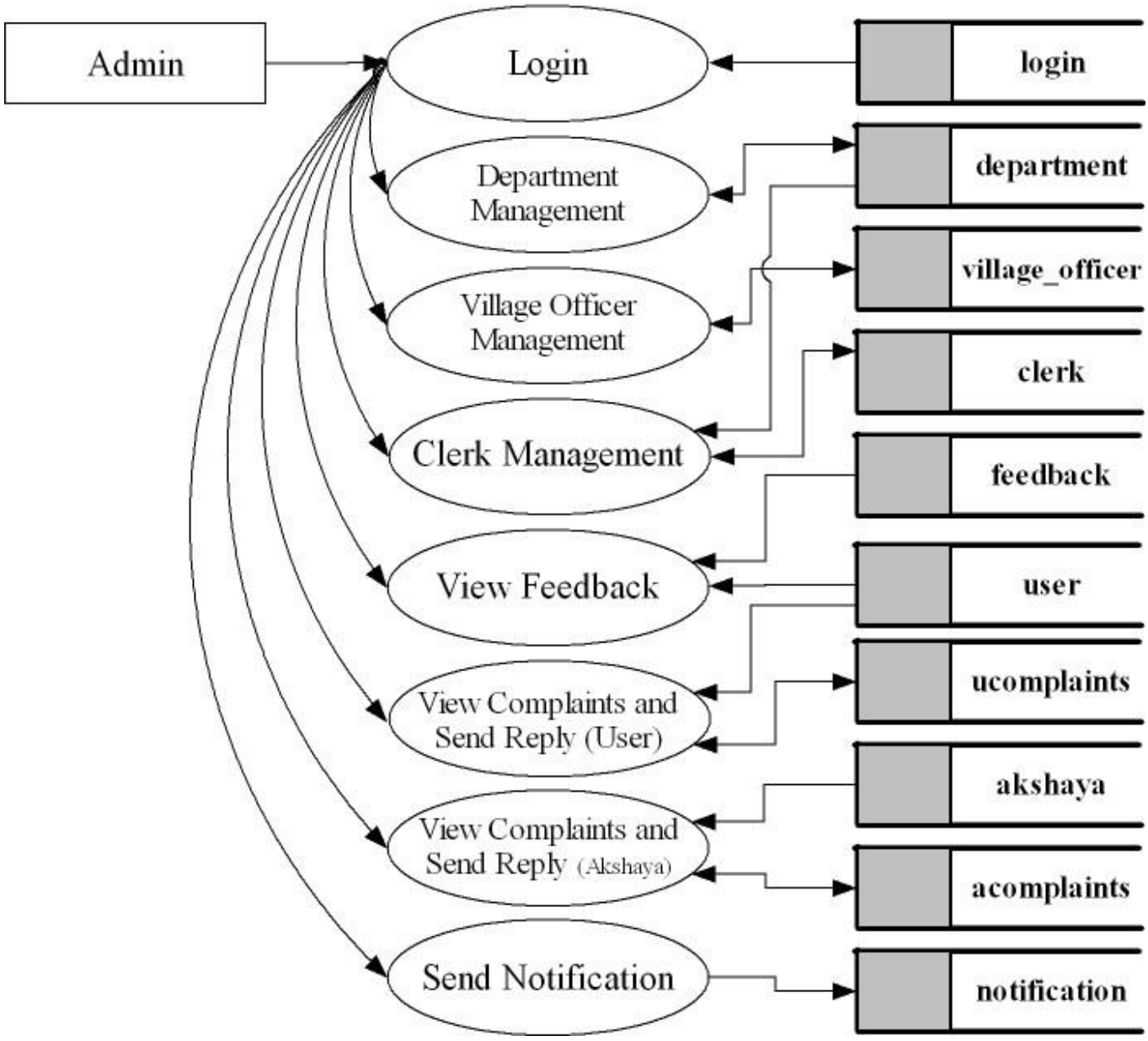
The DFD at the simplest level is referred to as the “CONTEXT ANALYSIS DIAGRAM”. These are expanded by level. Each explains its process in detail. Processors are numbered for easy identification and are normally labelled in block letters. Each data flow is labelled for easy understanding.

LEVEL 0

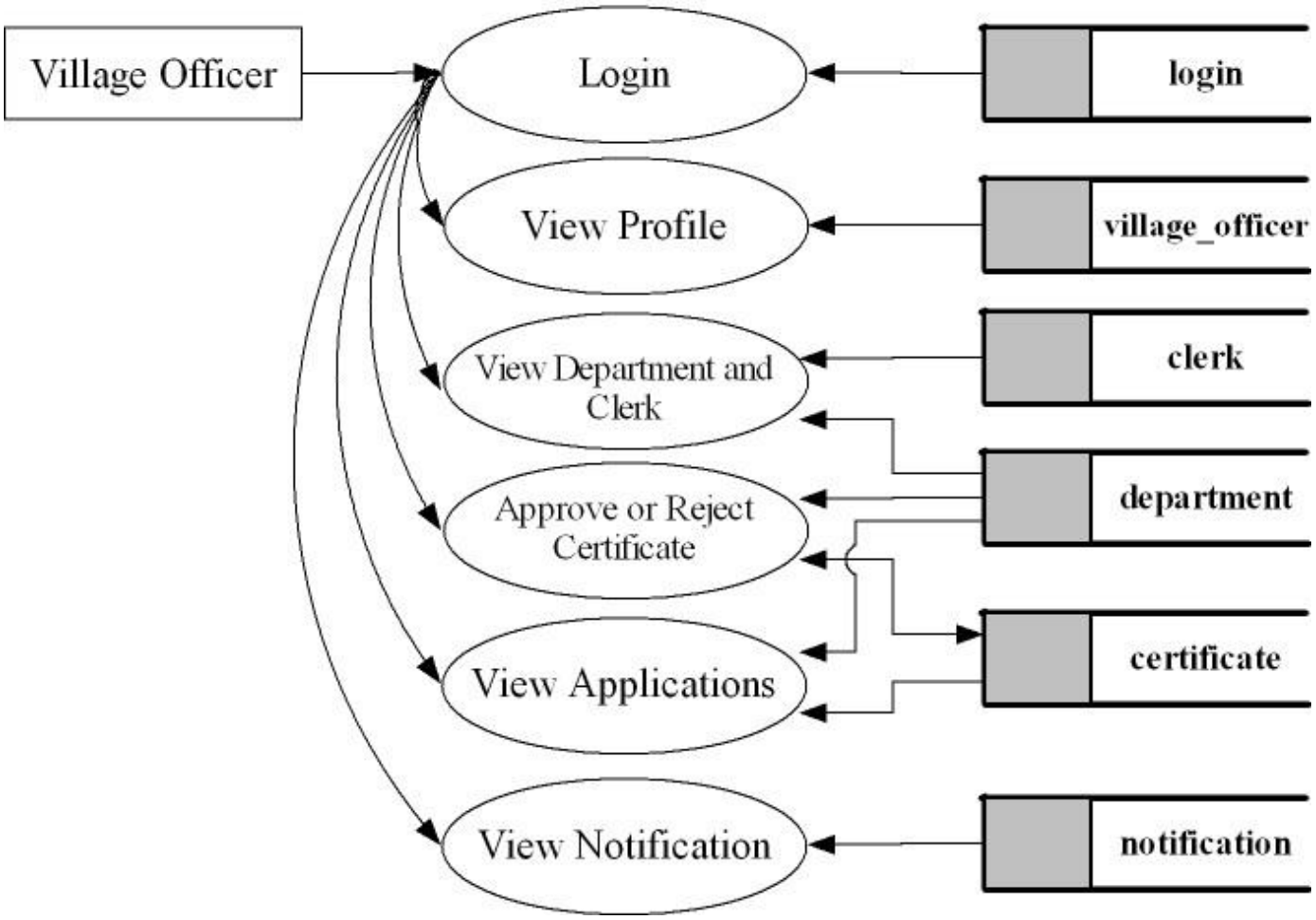
LEVEL 1



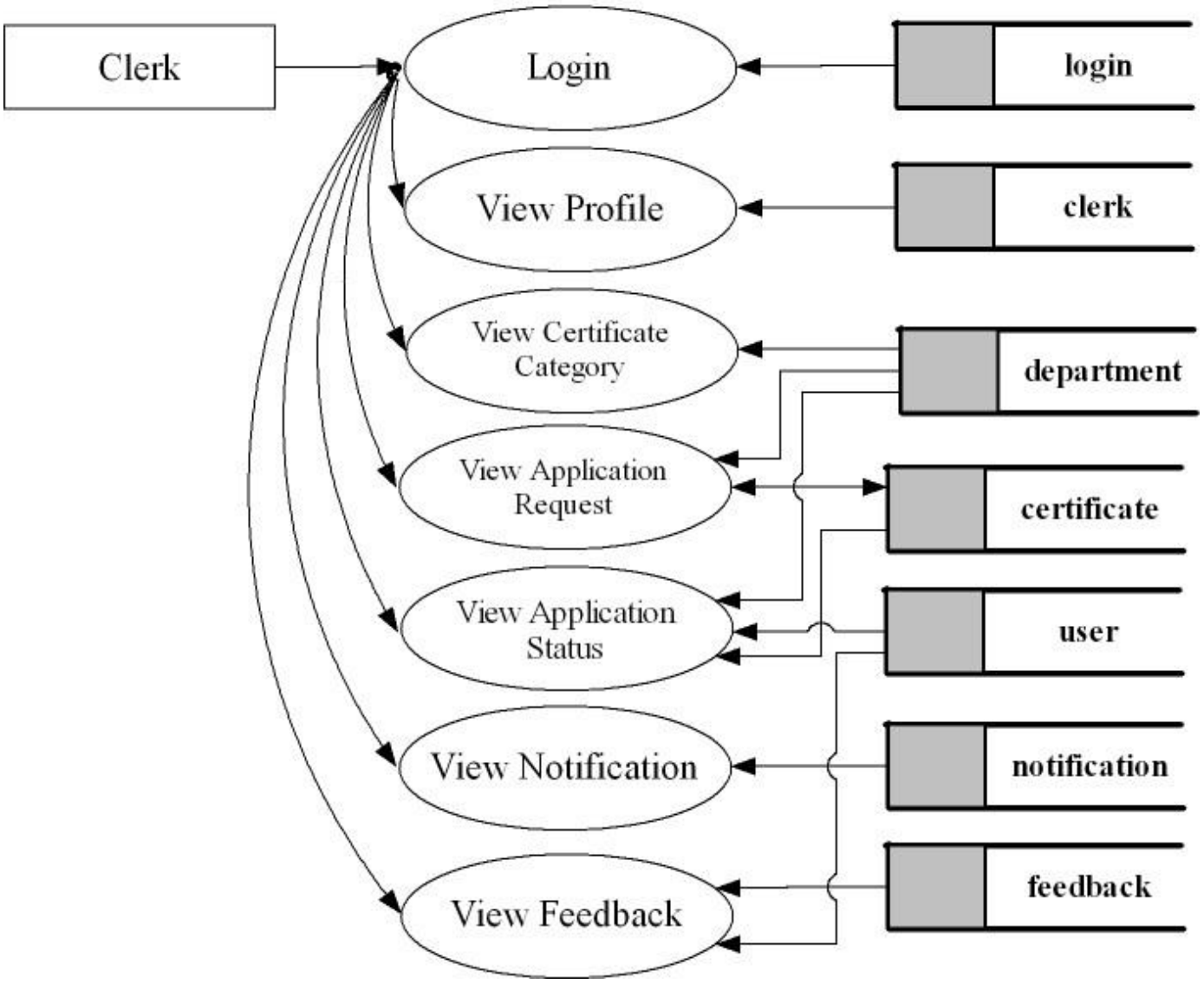
LEVEL 1.1


Admin
Login
Department Management
Village Officer Management
Clerk Management
View Feedback
View Complaints and Send Reply (User)
View Complaints and Send Reply (Akshaya)
Send Notification
login
department
village_officer
clerk
feedback
user
ucomplaints
akshaya
acomplaints
notification

LEVEL 1.2

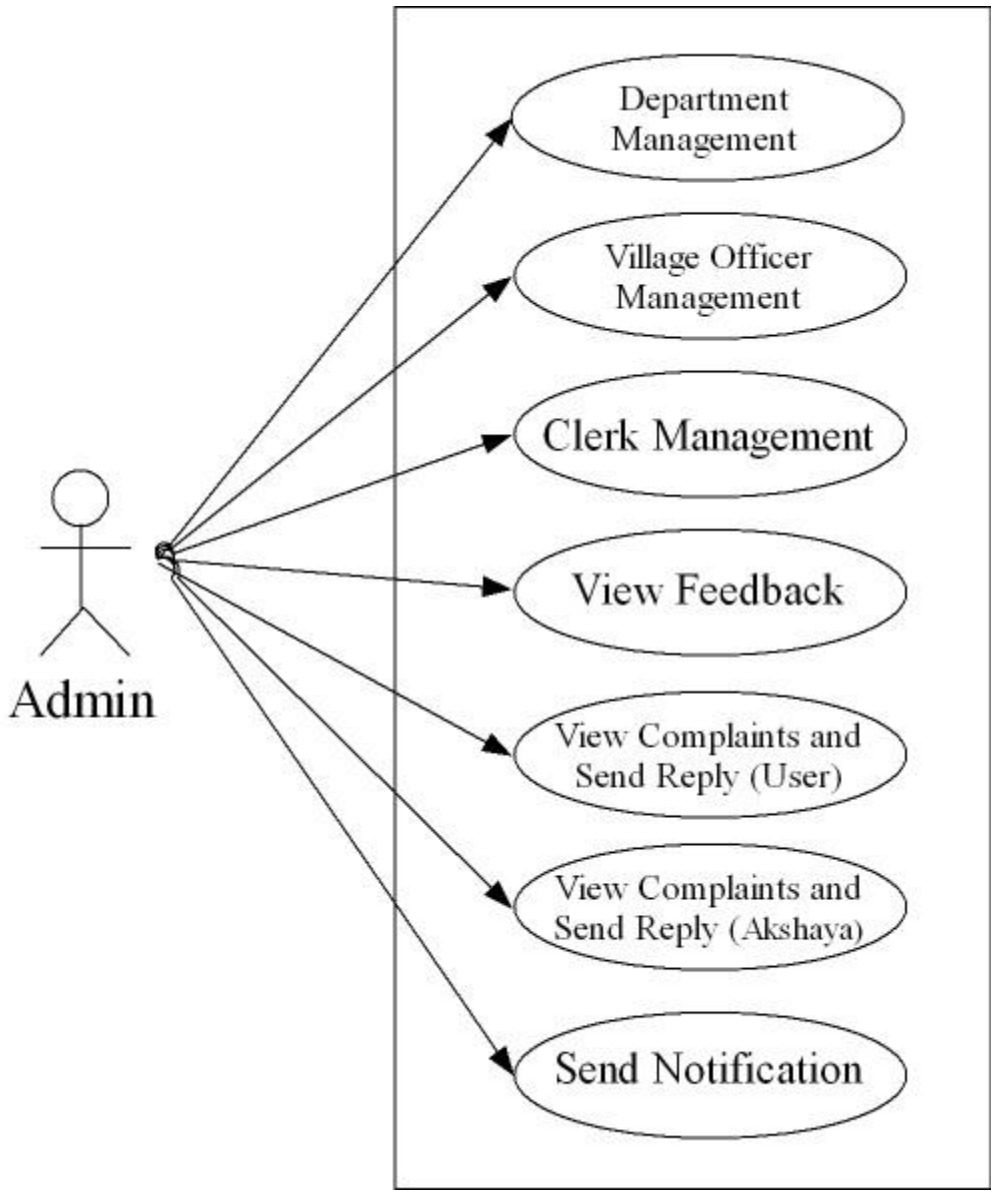

Village Officer
Login
View Profile
View Department and Clerk
Approve or Reject Certificate
View Applications
View Notification
login
village_officer
clerk
department
certificate
notification

LEVEL 1.3

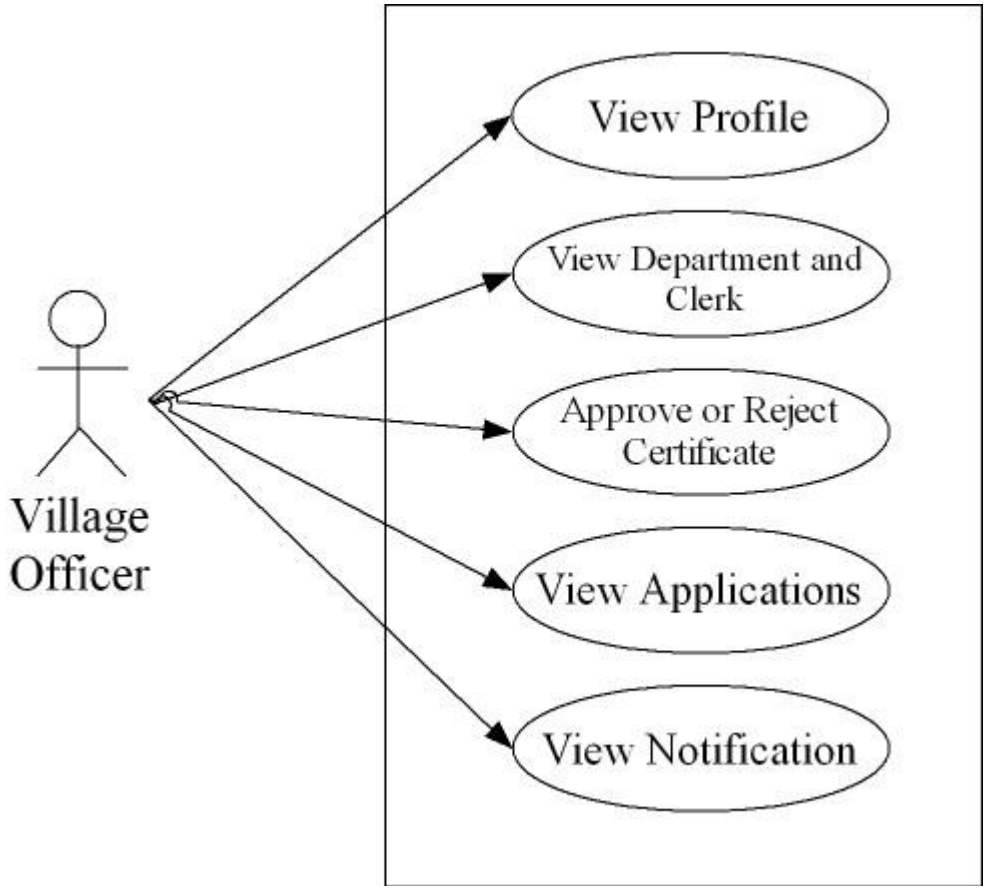

Clerk
Login
View Profile
View Certificate Category
View Application Request
View Application Status
View Notification
View Feedback
login
clerk
department
certificate
user
notification
feedback

USECASE DIAGRAM:

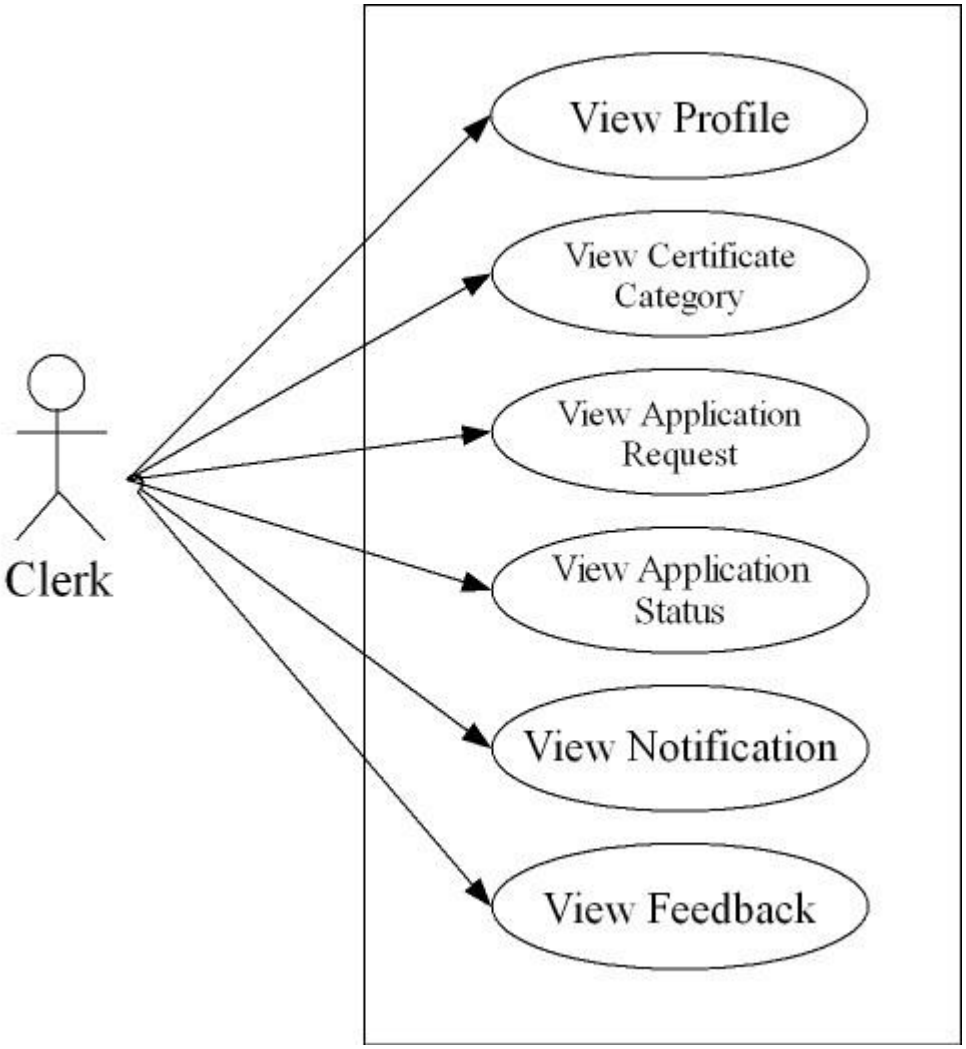
A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.



- Village Officer



- Clerk


View Profile
View Certificate Category
View Application Request
View Application Status
View Notification
View Feedback
Clerk

ER DIAGRAM:

An Entity-Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people,objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, educationand research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

The components and features of an ER diagram:

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. Here’s a glossary:

Entity:

A definable thing such as a person, object, concept or event that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.

Entity type: A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars or products.
Entity set: Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day.
Entity keys: Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate or primary.
Super key: A set of attributes (one or more) that together define an entity in an entity set.
Candidate key: A minimal super key. It has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key.
Primary key: A candidate key chosen by the database designer to uniquely identify the entity set.
Foreign key: Identifies the relationship between entities.

Relationship:

How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.

Attribute:

A property or characteristic of an entity. Often shown as an oval or circle.

Descriptive attribute: A property or characteristic of a relationship (versus of an entity)

Attribute categories: Attributes are categorized as simple, composite, derived, as well as single-value or multi-value. Simple: This means the attribute value is atomic and can't be further divided, such as a phone number.

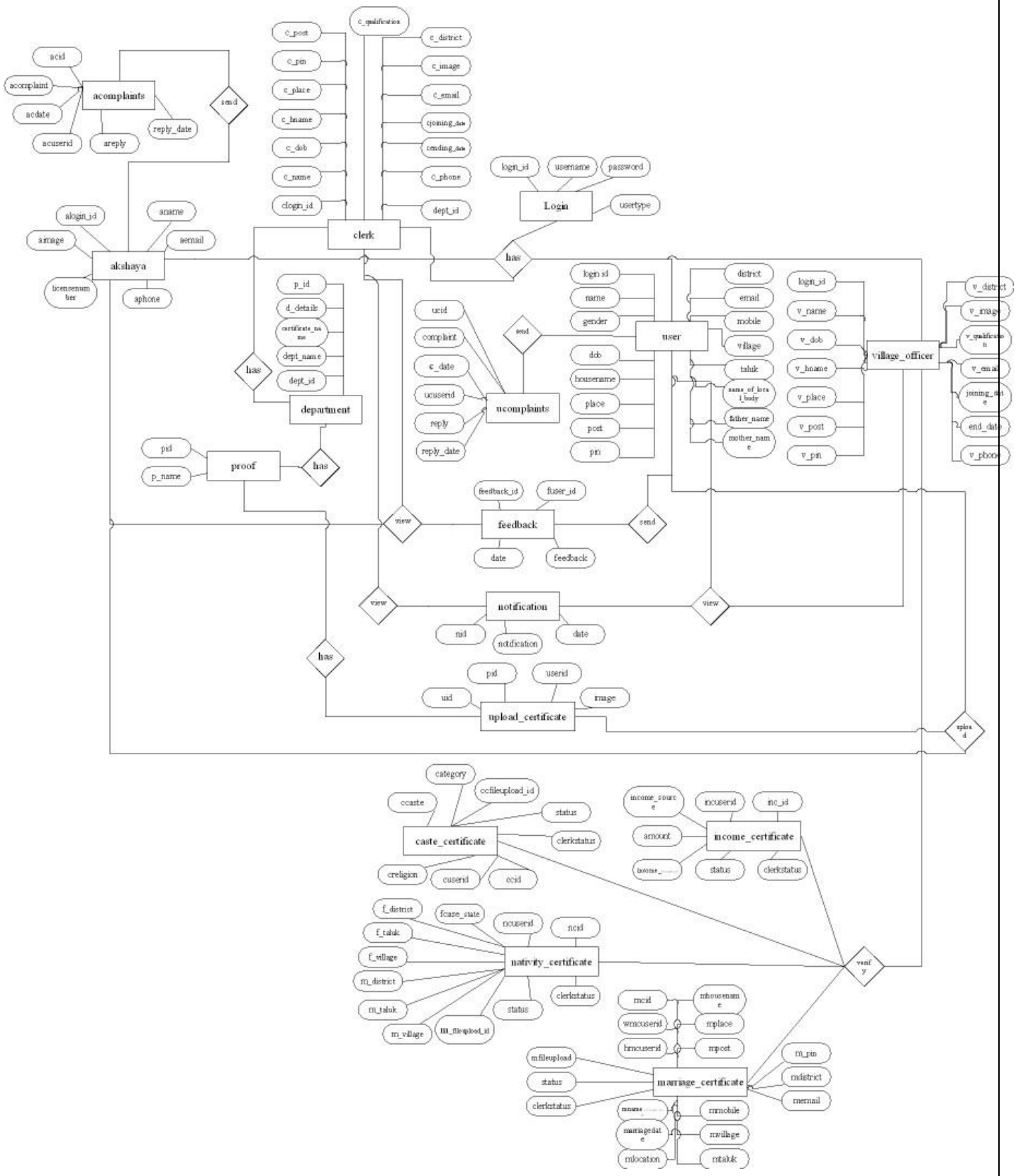
Composite: Sub-attributes spring from an attribute.

Derived: Attributed is calculated or otherwise derived from another attribute, such as age from a birth date.

Multi-value: More than one attribute value is denoted, such as multiple phone numbers for a person.

Single-value: Just one attribute value. The types can be combined, such as simple single-value attributes orcomposite multi-value attributes.

ER Diagram


acid
acomplaint
acdate
acuserid
areply
reply_date
acomplaints
send
c_qualification
c_post
c_pin
c_place
c_lname
c_dob
c_name
clogin_id
c_district
c_image
c_email
cjoining_date
cending_date
c_phone
dept_id
clerk
login_id
username
password
Login
usertype
has
alogin_id
aname
aimage
aemail
akshaya
licensenumber
aphone
p_id
d_details
certificate_name
dept_name
dept_id
department
ucid
complaint
c_date
ucuserid
reply
reply_date
ucomplaints
send
login_id
name
gender
dob
housename
place
post
pin
user
district
email
mobile
village
taluk
name_of_local_body
father_name
mother_name
login_id
v_name
v_dob
v_lname
v_place
v_post
v_pin
village_officer
v_district
v_image
v_qualification
v_email
joining_date
end_date
v_phone
pid
p_name
proof
has
feedback_id
fuser_id
feedback
date
feedback
view
send
view
notification
nid
notification
date
view
has
pid
userid
uid
image
upload_certificate
upload
category
ocfileupload_id
ccaste
status
caste_certificate
clerkstatus
creligion
cuserid
ccid
income_source
incuserid
inc_id
amount
income_certificate
income_
status
clerkstatus
f_district
fcaste_state
ncuserid
ncid
f_taluk
f_village
nativity_certificate
m_district
m_taluk
m_village
m_fileupload_id
status
clerkstatus
verify
rncid
rnhousename
rnplace
rnpost
wncuserid
hncuserid
mfileupload
status
clerkstatus
marriage_certificate
m_pin
mdistrict
memail
rnmobile
rnvillage
rntaluk
marriagedate
mlocation

User Interface Design

User interface (UI) design or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal ofuser interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centred design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements to create a system that is notonly operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects, from computer systems to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centred on their expertise, whether it is software design, user research, web design, or industrial design.

User interface design requires a good understanding of user needs. It mainly focuses on the needs of the platform and its user expectations. There are several phases and processes in the user interface design, some of which are more demanded upon than others, depending on the project.

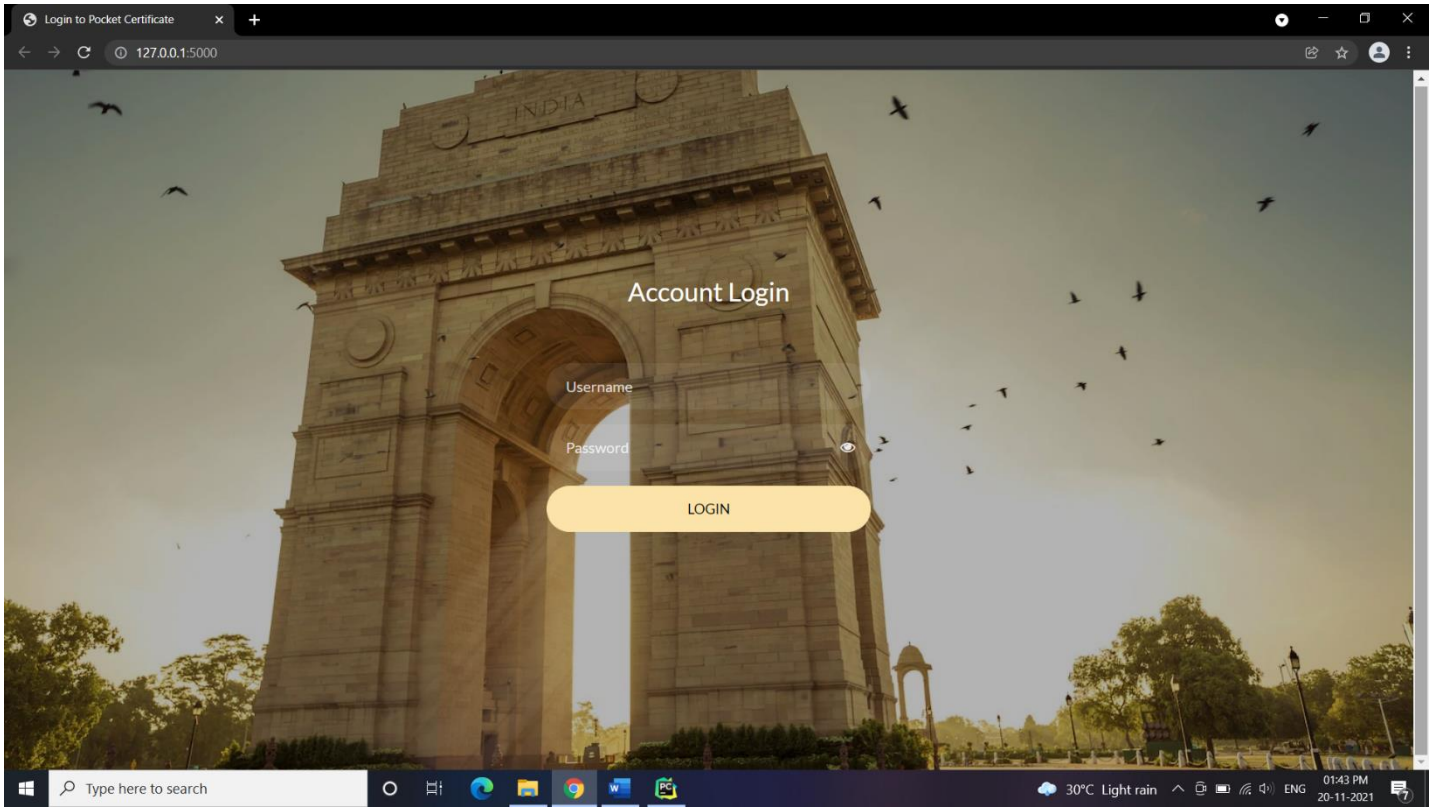
- Functionality requirements gathering – assembling a list of the functionality required by the system toaccomplish the goals of the project and the potential needs of the users.
- User and task analysis – a form of field research, it's the analysis of the potential users of the system by studying how they perform the tasks that the design must support, and conducting interviews to elaborate their goals. Typical questions involve:
    - What would the user want the system to do?
    - How would the system fit in with the user's normal workflow or daily activities?
    - How technically savvy is the user and what similar systems does the user already use?
    - What interface look & feel styles appeal to the user?

- Usability inspection – letting an evaluator inspect a user interface. This is generally considered to be cheaper to implement than usability testing (see step below) and can be used early on in the development process since it can be used to evaluate prototypes or specifications for the system,

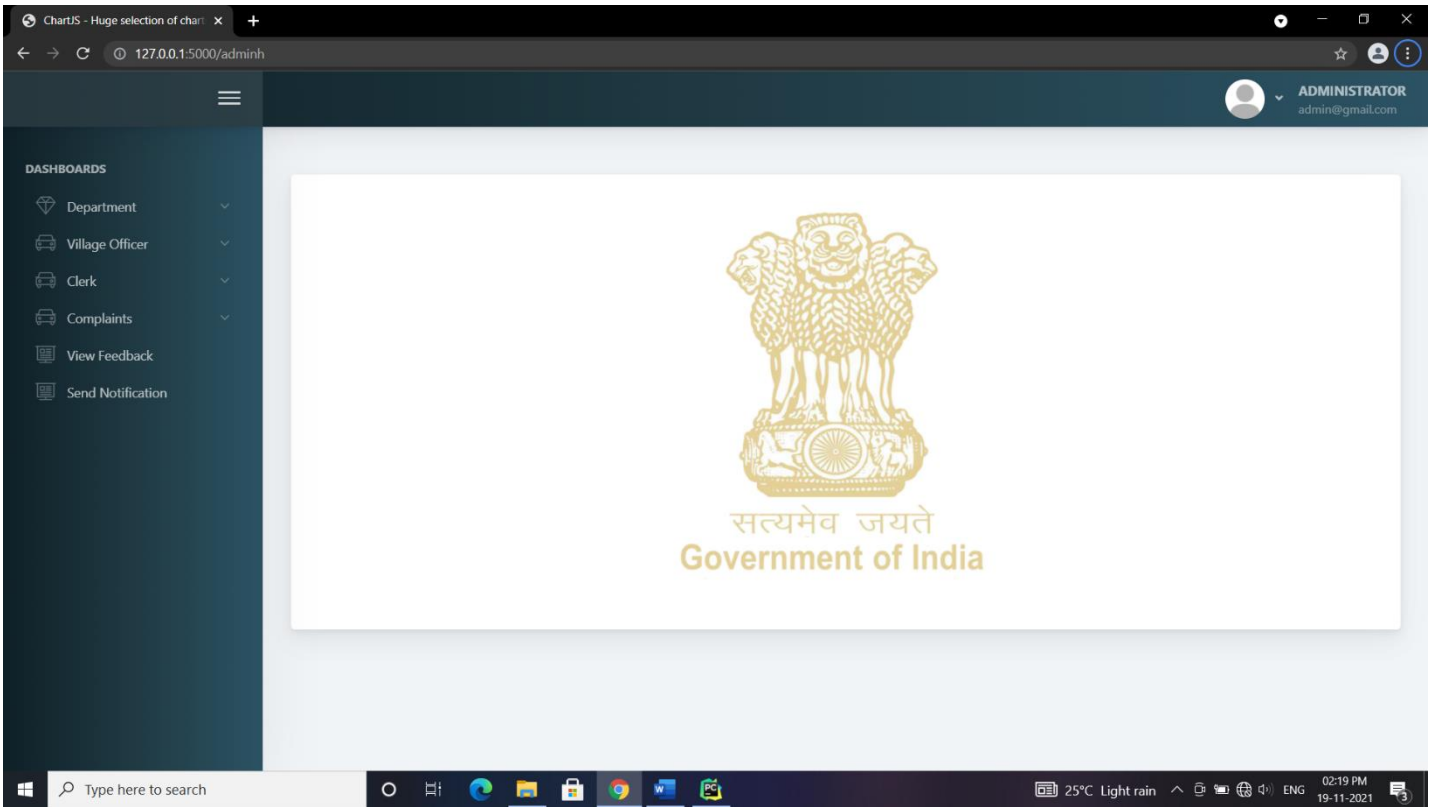
which usually cannot be tested on users. Some common usability inspection methods include a cognitive walkthrough, which focuses the simplicity to accomplish tasks with the system for new users, heuristicevaluation, in which a set of heuristics are used to identify usability problems in the UI design, and pluralistic walkthrough, in which a selected group of people step through a task scenario and discussusability issues.

- Usability testing – testing of the prototypes on an actual user—often using a technique called think aloud protocol where you ask the user to talk about their thoughts during the experience. User interface design testing allows the designer to understand the reception of the design from the viewer's standpoint, and thus facilitates creating successful applications.
- Graphical user interface design – actual look and feel the design of the final graphical user interface (GUI). These are design's control panels and faces; voice-controlled interfaces involve oral-auditory interaction, while gesture-based interfaces witness users engaging with 3D design spaces via bodily motions. It may be based on the findings developed during the user research and refined to fix any usability problems found through the results of testing. Depending on the type of interface being created,this process typically involves some computer programming in order to validate forms, establish links or perform a desired action.

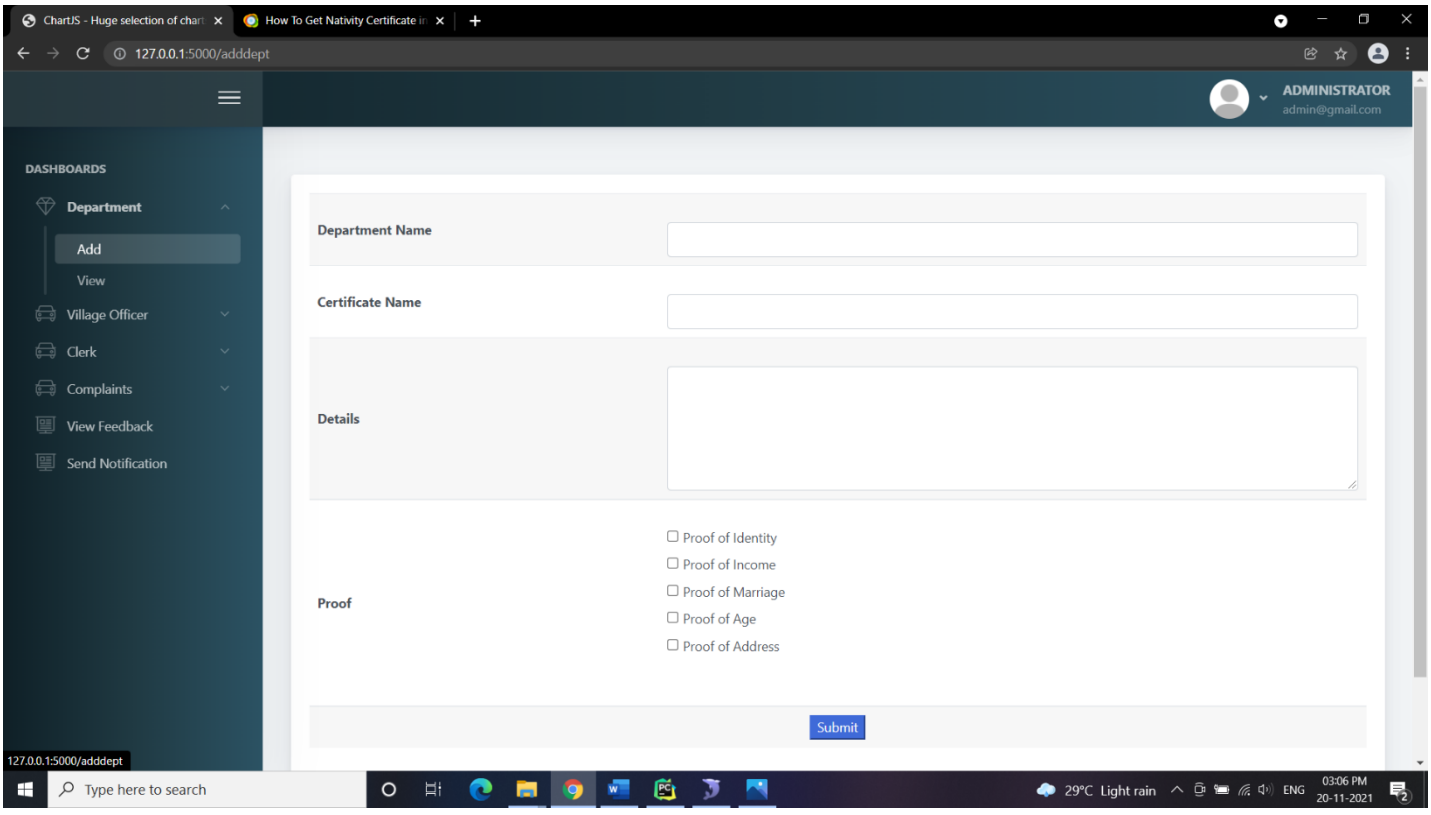
Our project has many validation forms which have a simple and easy to use interface design.
admin is our first module that is a webpage created using python and HTML, when we open this web page, the login page will appear as shown below:



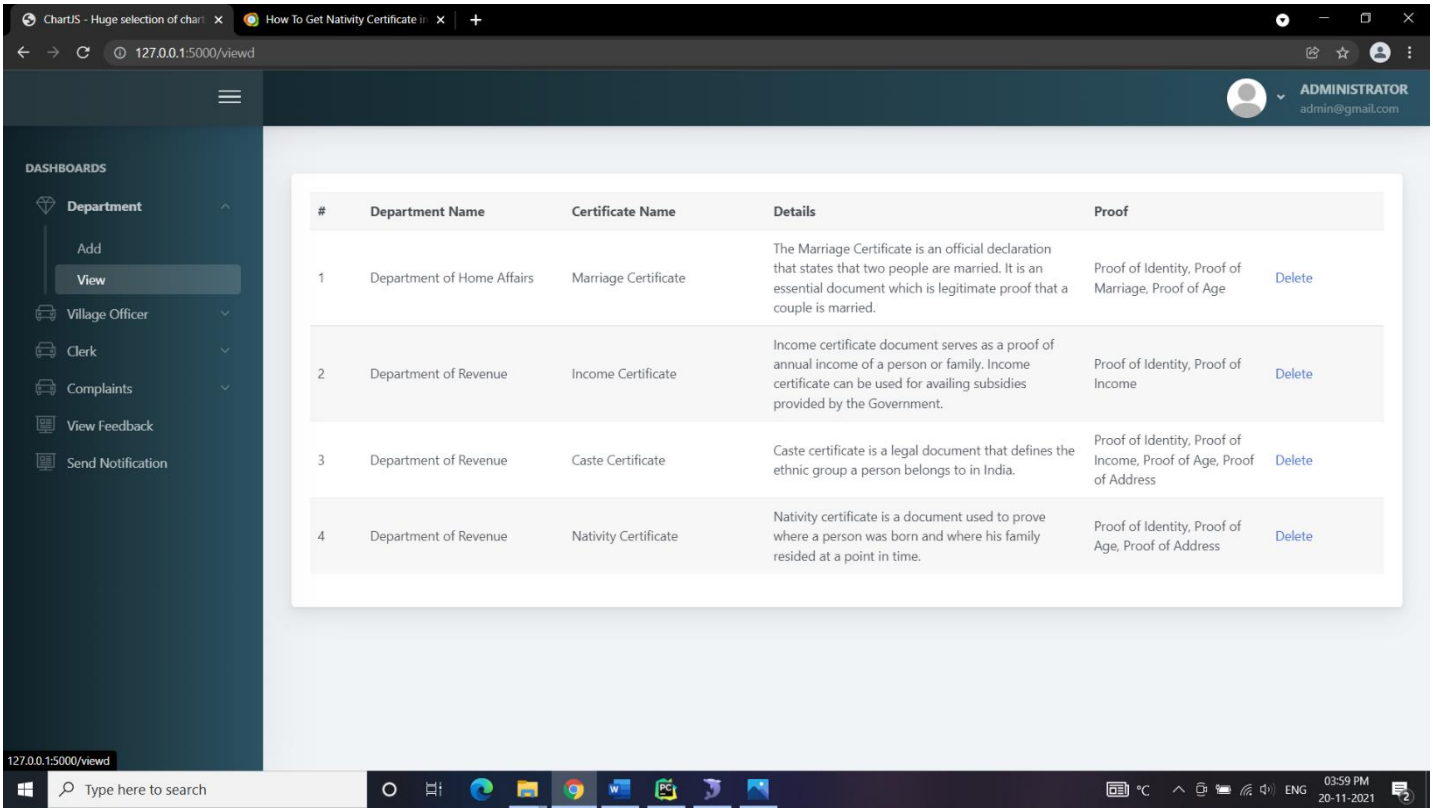
If we log in to this webpage we get an interface like this:



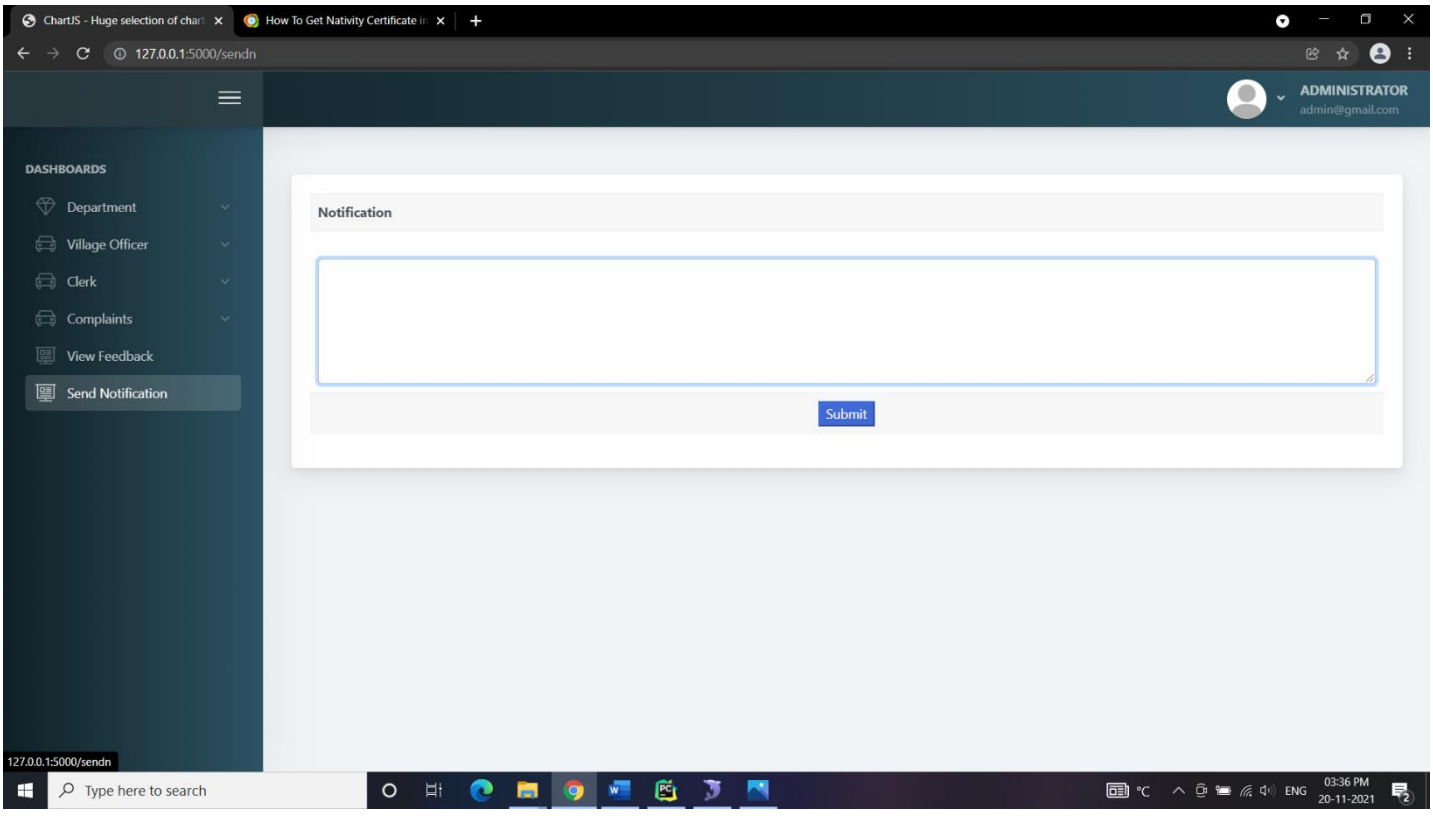
On this page the Admin can add a Department:



On this page the Admin can view and delete the Departments:



Admin can also send notifications to the Village Officer and Clerk from this webpage:



5.System Testing & Implementation

Testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is a series of different tests whose sole purpose is to exercise the full computer-based system.

System Testing involves testing the software code for following

- Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
- Verify thorough testing of every input in the application to check for desired outputs.
- Testing of the user's experience with the application.

That is a very basic description of what is involved in system testing. You need to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

As with almost any software engineering process, software testing has a prescribed order in which things should be done. The following is a list of software testing categories arranged in chronological order. These are the steps taken to fully test new software in preparation for marketing it:

- Unit testing performed on each module or block of code during development. Unit Testing is normally done by the programmer who writes the code.

- Integration testing done before, during and after integration of a new module into the main software package. This involves testing each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.

- System testing done by a professional testing agent on the completed software product before it is introduced to the market.

- Acceptance testing - beta testing of the product done by the actual end-users.

5.1 Testing Approaches

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred

to as dynamic testing.

Static testing is often implicit, as proofreading, plus when programming tools/text editors checksource code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for these are either using stubs/drivers or execution from a debugger environment.

Static testing involves verification, whereas dynamic testing also involves validation.

Passive testing means verifying the system behaviour without any interaction with the software product. Contrary to active testing, testers do not provide any test data but look at system logs and traces. They mine for patterns and specific behaviour in order to make some kind of decision. This is related to offline runtime verification and logs analysis.

5.1.1 Unit Testing:

In computer programming, unit testing is a software testing method by which individual units of source code—sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures—are tested to determine whether they are fit for use.

Unit tests are typically automated tests written and run by software developers to ensure that asection of an application (known as the "unit") meets its design and behaves as intended. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface,such as a class, but could be an individual method. By writing tests first for the smallest testable units, then the compound behaviours between those, one can build up comprehensive tests for complex applications.

To isolate issues that may arise, each test case should be tested independently. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist in testing a module in isolation.

During development, a software developer may code criteria, or results that are known to be good, into the test to verify the unit's correctness. During test case execution, frameworks log tests that failany criterion and report them in a summary. For this, the most commonly used approach is test-function - expected value.

Writing and maintaining unit tests can be made faster by using parameterized tests. These allow the execution of one test multiple times with different input sets, thus reducing test code duplication. Unlike traditional unit tests, which are usually closed methods and test invariant conditions,parameterized tests take any set of parameters. Parameterized tests are supported by Testing, JUnit and its .Net counterpart, XUnit. Suitable parameters for the unit tests may be supplied manually or in some cases are automatically

generated by the test framework. In recent years' support was added for writing more powerful (unit) tests, leveraging the concept of theories,test cases that execute the same steps, but using test data generated at runtime, unlike regular parameterized tests that use the same execution steps with input sets that are pre-defined.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

Unit testing finds problems early in the development cycle. This includes both bugs in the programmer's implementation and flaws or missing parts of the specification for the unit. The process of writing a thorough set of tests forces the author to think through inputs, outputs, and errorconditions, and thus more crisply define the unit's desired behaviour. The cost of finding a bug before coding begins or when the code is first written is considerably lower than the cost of detecting, identifying, and correcting the bug later. Bugs in released code may also cause costly problems for the end-users of the software. Code can be impossible or difficult to unit test if poorly written, thus unit testing can force developers to structure functions and objects in better ways.

In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since theunit tests alert the development team of the problem before handing the code off to testers or clients,potential problems are caught early in the development process.

Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write testcases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes that may break a design contract.

Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API).

Unit test cases embody characteristics that are critical to the success of the unit. These characteristics

can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, althoughmany software development environments do not rely solely upon code to document the product in development.

When software is developed using a test-driven approach, the combination of writing the unit test to specify the interface plus the refactoring activities performed after the test has passed may take the place of formal design. Each unit test can be seen as a design element specifying classes, methods, and observable behaviour.

5.1.2 Integration Testing:

System integration testing (SIT) involves the overall testing of a complete system of many subsystem components or elements. The system under test may be composed of hardware or software, or hardware with embedded software, or hardware/software with human-in-the-loop testing.

SIT consists, initially, of the "process of assembling the constituent parts of a system in a logical, cost-effective way, comprehensively checking system execution (all nominal & exceptional paths), and including a full functional check-out. Following integration, a system test is a process of "verifying that the system meets its requirements, and validating that the system performs in accordance with the customer or user expectations.

In technology product development, the beginning of system integration testing is often the first time that an entire system has been assembled such that it can be tested as a whole. In order to make system testing most productive, the many constituent assemblies and subsystems will have typically gone through a subsystem test and successfully verified that each subsystem meets its requirements at the subsystem interface level.

In the context of software systems and software engineering, system integration testing is a testing process that exercises a software system's coexistence with others. With multiple integrated systems, assuming that each has already passed system testing, SIT proceeds to test their required interactions. Following this, the deliverables are passed on to acceptance testing or software SIT is part of the software testing life cycle for collaborative projects. Usually, a round of SIT precedes the user acceptance test (UAT) round. Software providers usually run a pre-SIT round of tests beforeconsumers run their SIT test cases.

For example, if an integrator (company) is providing an enhancement to a customer's existing solution, then they integrate the new application layer and the new database layer with the customer'sexisting application and database layers. After the integration is complete, users use both the new part (extended part) and the old part (pre-existing part) of the integrated application to update data. A process should

exist to exchange data imports and exports between the two data layers. This data exchange process should keep both systems up-to-date. The purpose of system integration testing is to ensure all parts of these systems successfully co-exist and exchange data where necessary.

There may be more parties in the integration, for example, the primary customer (consumer) can have their own customers; there may be also multiple providers.

5.1.3 Validation Testing

Software validation checks that the software product satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements, not as specification artefacts or as needs of those who will operate the software only; but, as the needs of all the stakeholders (such as users, operators, administrators, managers, investors, etc.). There are two ways to perform software validation: internal and external. During internal software validation, it is assumed that the goals of the stakeholders were correctly understood and that they were expressed in the requirement artefactsprecisely and comprehensively. If the software meets the requirement specification, it has been internally validated. External validation happens when it is performed by asking the stakeholders if the software meets their needs. Different software development methodologies call for different levels of user and stakeholder involvement and feedback; so, external validation can be a discrete ora continuous event. Successful final external validation occurs when all the stakeholders accept the software product and express that it satisfies their needs. Such final external validation requires the use of an acceptance test which is a dynamic test.

However, it is also possible to perform internal static tests to find out if it meets the requirements specification but that falls into the scope of static verification because the software is not running.

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product indeed meets the client's needs. It can also be defined as demonstrating that the product fulfils its intended use when deployed in an appropriate environment.

Output testing:

After performing the validation testing, the next step is the output testing of the enhanced system. No system could be useful if it does not produce the required output in the required format. The outputs generated or displayed by the system are tested asking the users about the format required by them. During output testing, the user required some new report layout changes. These changes were done andtested before the final delivery of the system.

The "box" approach:

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that the tester takes when designing test cases. A hybrid approach called grey-box testing may also be applied to software testing methodology. Withthe concept of grey-box testing—which develops tests from specific design elements—gaining prominence, this "arbitrary distinction" between black and white-box testing has faded somewhat.

White-box Testing:

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testingnodes in a circuit, e.g., in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration, and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between unitsduring integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Black-box Testing:

Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, andspecification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case or leaves some parts of the program untested. This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

5.2 Implementation Approaches

The final and important phase in the system in the life cycle is the life cycle in the implementation of the new system. The term implementation has different meanings ranging from the conversion of a basic application to a complete replacement of a computer system. The procedure, however, is virtually the same. Implementation includes all those activities that take place to convert from the old system to the new.

The method of implementation and time scale to be adopted is found out initially. Next, the system is tested properly and at the same time users are trained in the new procedure. Proper implementation is essential to provide a reliable system to meet organization requirements. Successful implementation may not guarantee improvement in the organization using the new system. But it will prevent improper installation. The implementation involves the following things.

- Careful planning.
- Investigation of the system consideration.
- Design the method to achieve the changeover.
- Evaluation of change over method.

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the uses that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, and evaluation, of changeover methods. Apart from planning major tasks of preparing the implementations are education and training of users. The more complex system being implemented, the more involved will be the system analysis and the design effort required just for implementation, on implementation coordinating committee based on policies of individual organization has been appointed.

6. Results and Discussions

6.1 Test report

Test cases 1: Login

SL_NO	Test scenario	Expected result	Actual result	Status
1	Click on the login button with an invalid username	An error message will occur	Invalid username or password	Passed
2	Click on the login button with an invalid password	An error message will occur	Invalid username or password	Passed
3	Login button pressed with no entries in the textbox	An error message will occur	"Please fill out the field" message displayed.	Passed
4	Login button pressed with a valid username and password	Login successfully	Login successfully	Passed

Test cases 2: Department Management

SL_NO	Test scenario	Expected result	Actual result	Status
1	Click on the Department Add link to Add Department.	Display the form of add department.	Form displayed	Passed
2	Add a Department with corresponding details	Displays a message Department added successfully	Message displayed	Passed
3	Add Department without entering any field	Display error Message	"Please fill out the field" message displayed	Passed
4	Delete the corresponding values of a Department	Deletes the data entered into the database	Message displayed	Passed

Test cases 3: Village Officer Management

SL_NO	Test scenario	Expected result	Actual result	status
1	Click on the Add Village Officer link to Add village officer	Display the form to add the village officer	Form displayed	Passed
2	Add a village officer with corresponding details	Displays a message Village Officer Added Successfully	Message displayed	Passed
3	Add village officer without entering email or phone number incorrect format	Display error message	Message displayed	Passed
4	Add village officer without entering any field	Display error message	"Please fill out the field" message displayed	Passed
5	Update the corresponding values of a village officer	Stores the data entered into the database	Message displayed	Passed
6	Delete the corresponding values of a village officer	Deletes the data entered into the database	Message displayed	Passed

Test case 4: Notification

SL_NO	Test scenario	Expected result	Actual result	status
1	Click on the Send Notification link to enter a notification	Display the form to send notification	Form displayed	Passed
2	Enter the notification and click submit	Displays a message Notification Sent Successfully	Message displayed	Passed
3	Click submit without entering the notification	Display error message	"Please fill out the field" message displayed	Passed

6.2 User Documentation

Our project pocket certificate provides a full-featured graphical interface that helps you to create electronic documents. The first interface we come across is the login webpage where the user can login as an admin, village officer or clerk depending on the users' position of work where each entity is provided with a unique interface based on their working ground. To login to their respective field of work, the user is requested to enter their email and password followed by a login button. By clicking the login button, if the requested details are specified, each entity is given a unique interface to work on.

If the user is logged in as an admin, the user is brought to an admin home webpage where the admin can perform certain functions such as add and view departments, village officers and clerks. The first function of the page is department management, the admin can add and view departments by entering the department details. The component that comes after the department management is the village officer which has a similar function to that of department management. The admin can add, delete and view village officers by entering the details. The function succeeding the village officer is the clerk management which equally has the same function as the department and village officer. The admin can also view user feedbacks and user complaints and reply to the users and Akshaya as well. The bottom last function in the admin page is the notification where the admin can send a notification to other entities like village officer and clerk.

The second module that represents the next echelon is the Village Officer. The village officer is bought to an interface that is specifically designed for the village officer to work on. The first function is the view profile function. By clicking on the function, the screen displays information about the village officer. The next function view department allows the village officer to view the department name, details, and the field of work teach department works on. The next function that comes after the view department is the view application function. The village officer can approve or reject certificates depending on the integrity of the documents submitted by the user. Certificates that are approved by the village officer can be viewed in the succeeding function, the approved certificate function. And finally, any information or updation from the admin can be viewed in the notification function.

The third module, The clerk, has almost similar functions to that of the village officers. If the user is logged in as a clerk, the user is given a unique interface to work on as well. The first function allows the user to view the details about the clerk. View application bar allows the user to approve or reject any certificate depending on the integrity of the documents. The admin can also send notifications to the clerk which can be viewed by the clerk by accessing the view notification's function. At last, the clerk can view user feedbacks using the view feedback's function.

7. CONCLUSION

7.1 Conclusion

Our project titled **POCKET CERTIFICATE** came to be after a comprehensive look into the digitalization of Government-issued certificates. Our intention with this project was to develop a system that would make the management of certificates a lot easier. The present system raises a lot of hurdles for an average citizen that makes it much harder to avail the services to get these documents in an easy and secure way. With our system, we rid of these very hurdles and offer a solution that is a lot more convenient and feasible.

A long period was taken for researching the essential features for these services. After collecting the required information, careful planning was taken in designing the user interface. This web-based project is completely user friendly that can be used by anyone with the most basic knowledge in handling a webpage.

The software is developed with a modular approach. All modules in the system have been tested and everything works successfully. We have currently completed three modules that serve the Governmental side of the system. As our next action, we will be developing more modules to serve the Average Citizen's side of the system.

The system is well efficient and can easily interact with the client. Any modification or enhancement if required can be done to suit the requirement. We also intend to add an additional module for Akshaya to help those citizens that do not have access to the Internet or any other reasons that wouldn't let them do it on their own. Further modules can be added to it to add more features as needs arise.

7.2 Future Scope of the Project

The project currently works on web browsers like Chrome, Microsoft Edge on Windows. A future enhancement may include supporting other Operating systems. The Citizen side can be developed to function on Android OS.

In our project, so far there consist only 3 modules. This project will be expanded upon further with modules for Citizens and Akshaya.

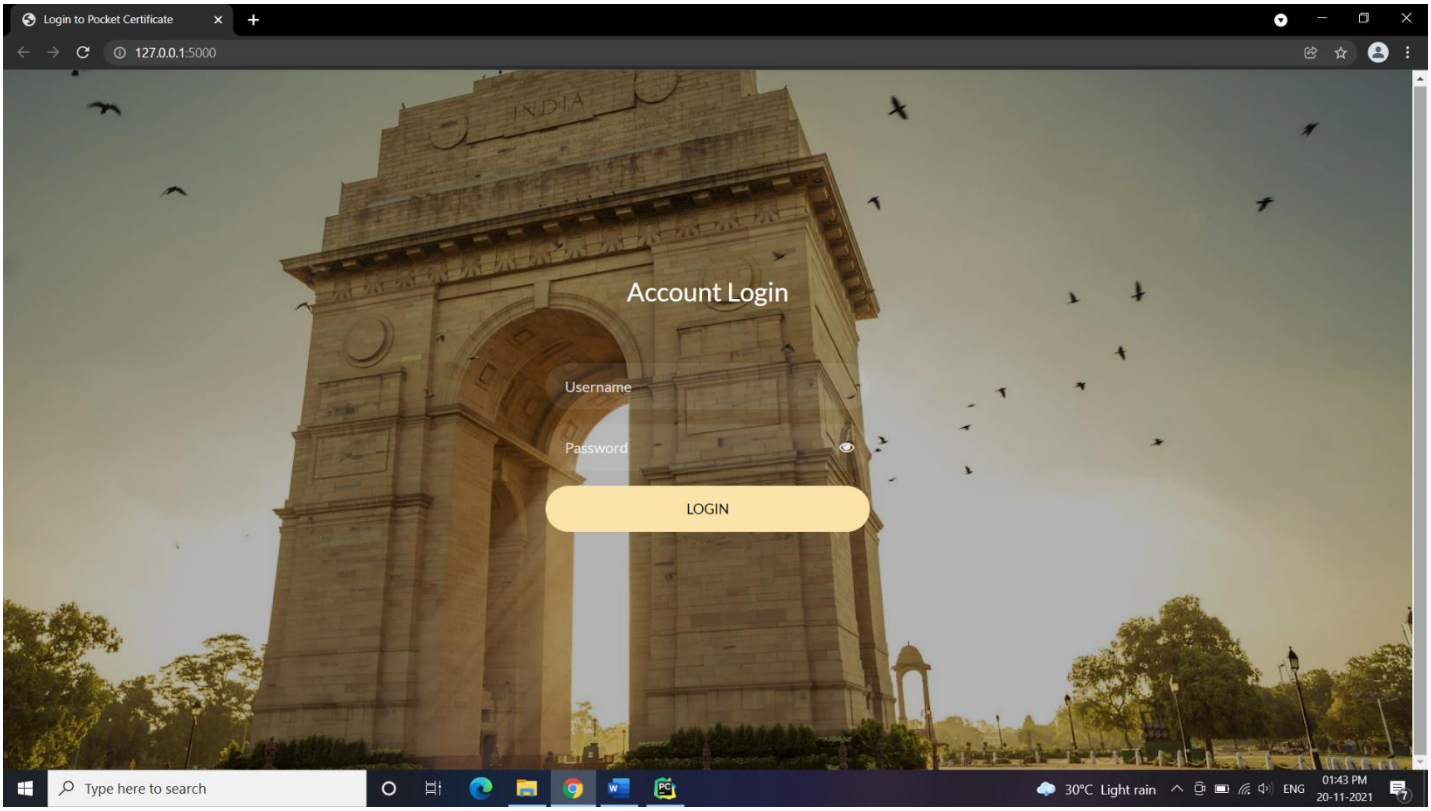
In the future, we also desire to expand our services to include more documents like School Certificates, Birth Certificates, Death Certificates and so on.

References

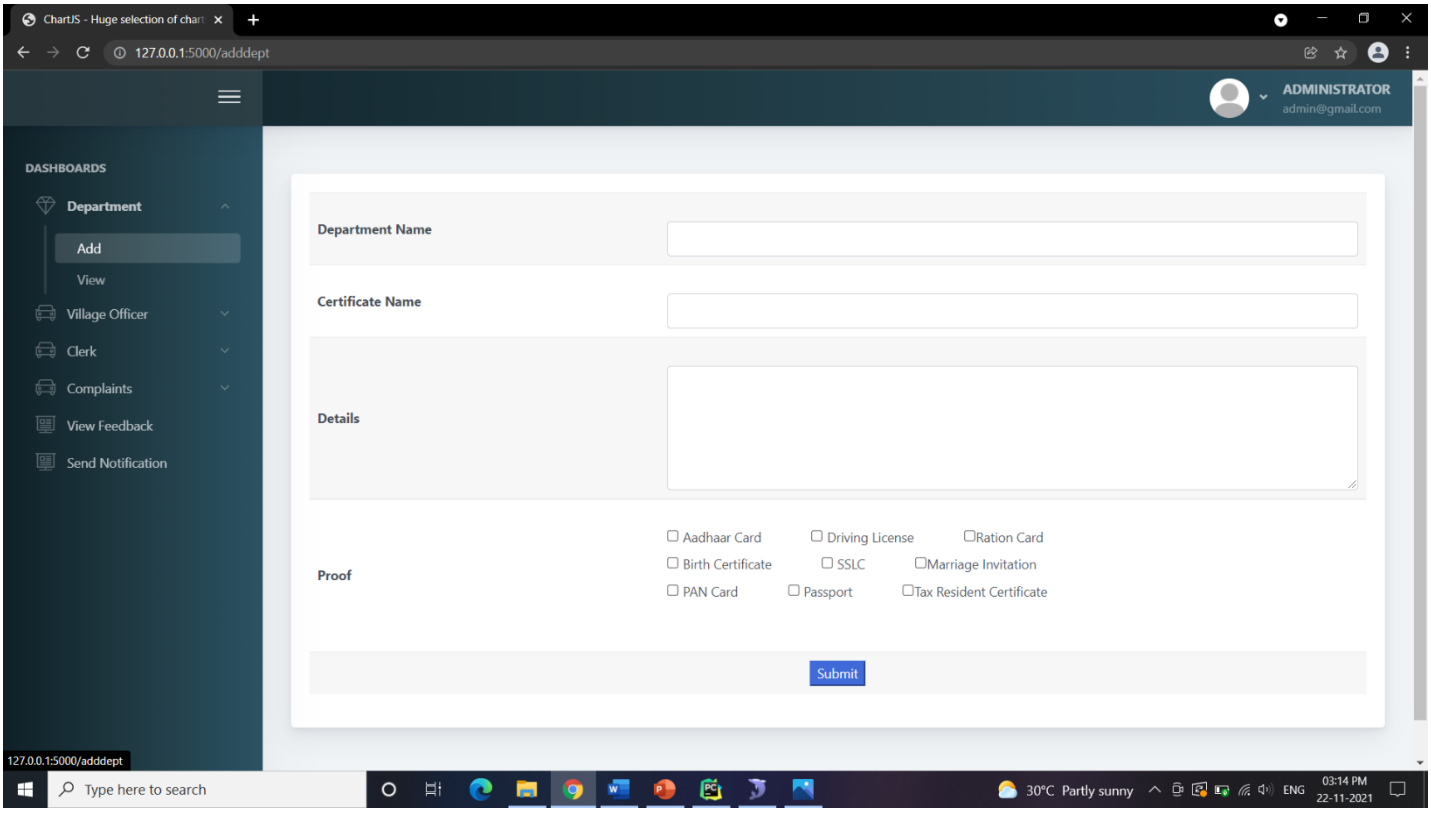
- www.w3schools.com
- www.geeksforgeeks.org
- www.javatpoint.com
- Software Engineering (Third Edition)
  – K. K. Aggarwal, Yogesh Singh
- Fundamentals of Python (First Programs)
  – Kenneth A. Lambert

Appendices

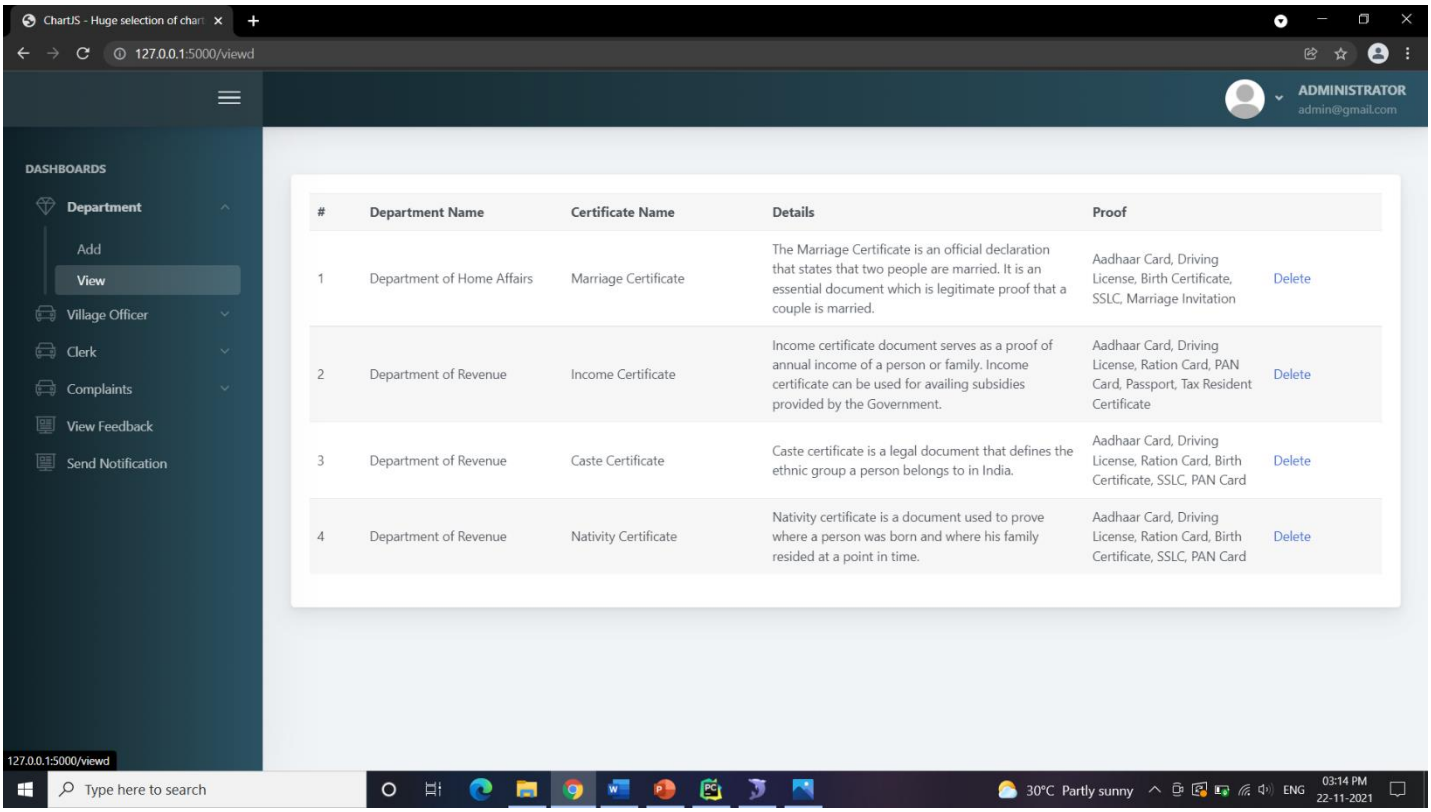
Webpage Login



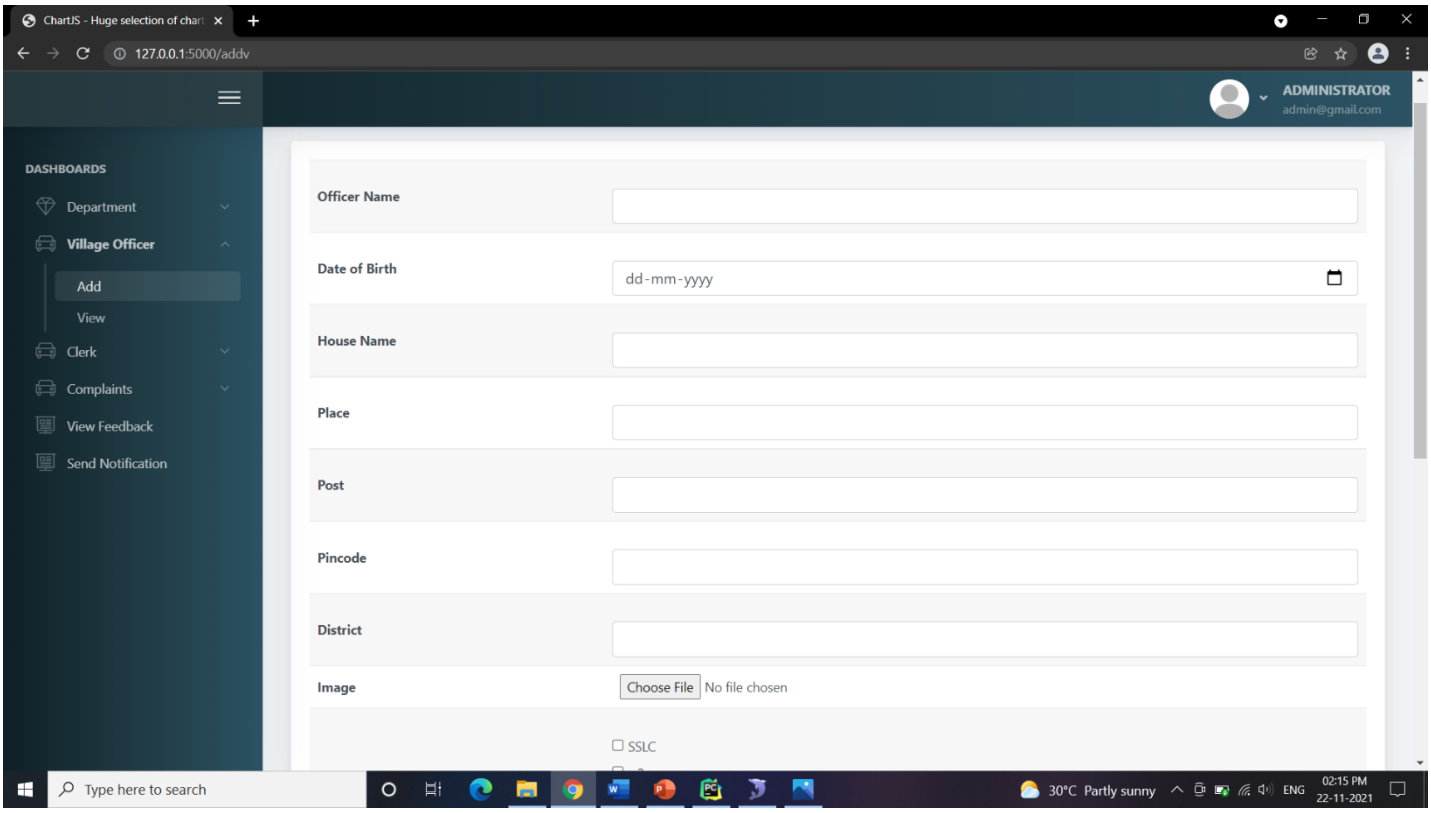
Add Department



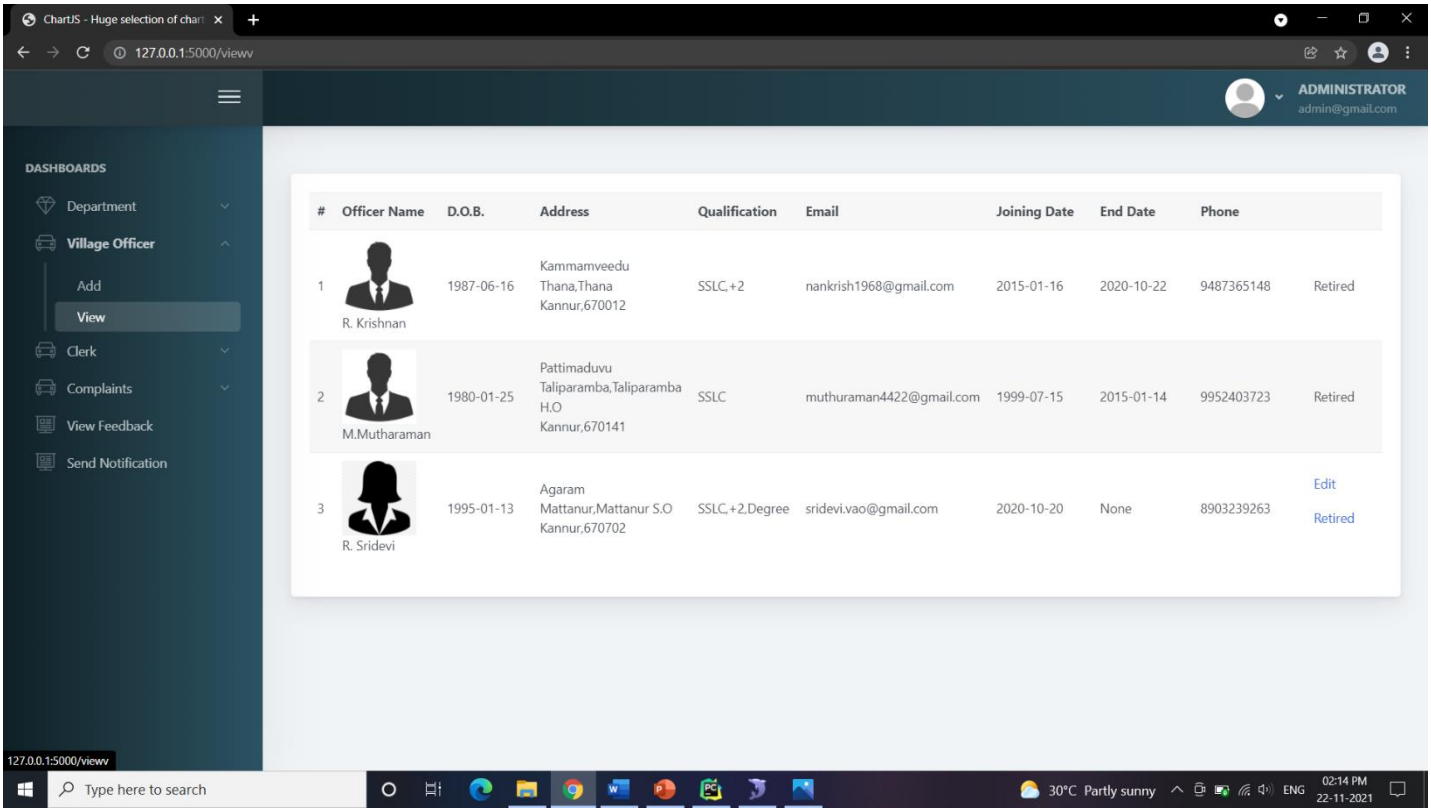
View Department



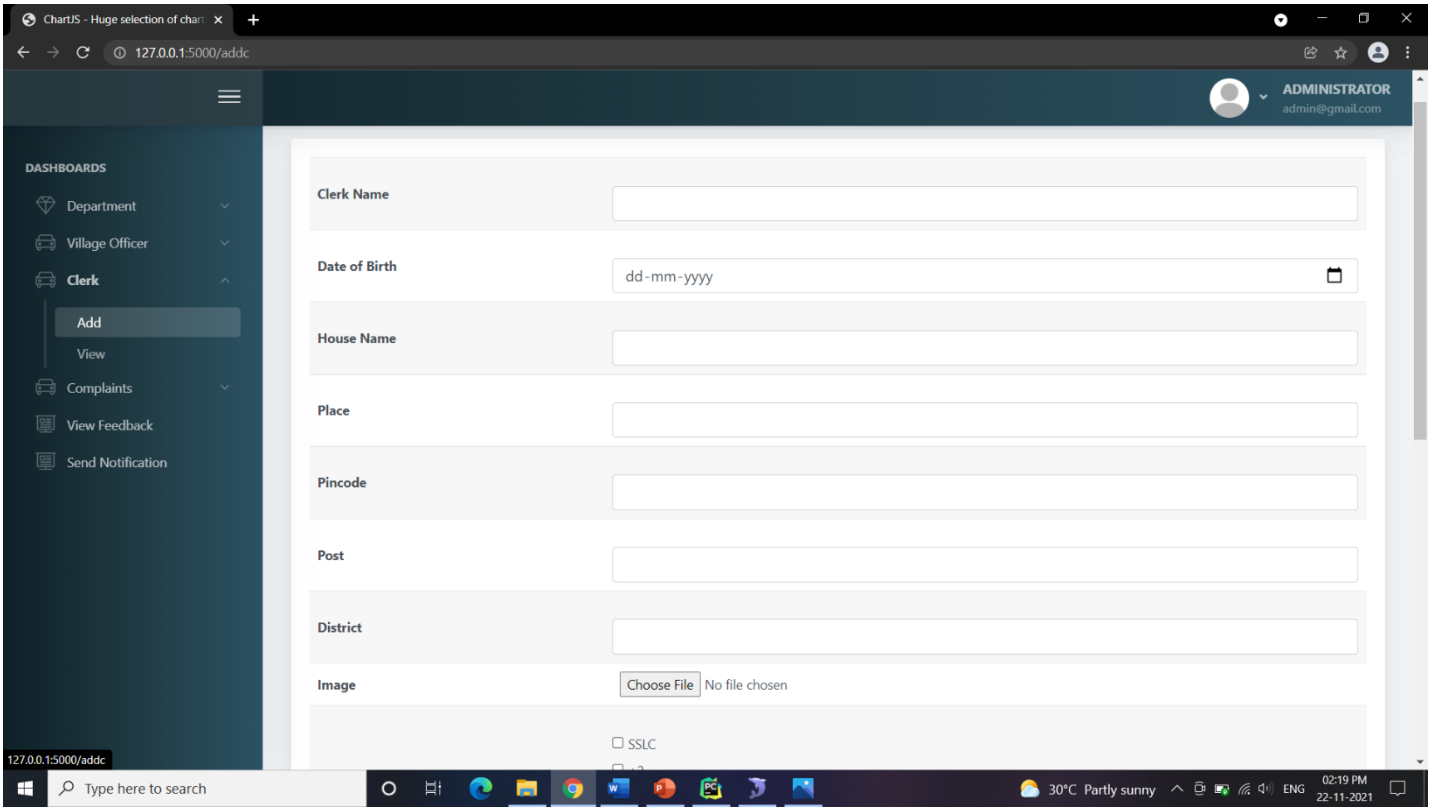
Add Village Officer



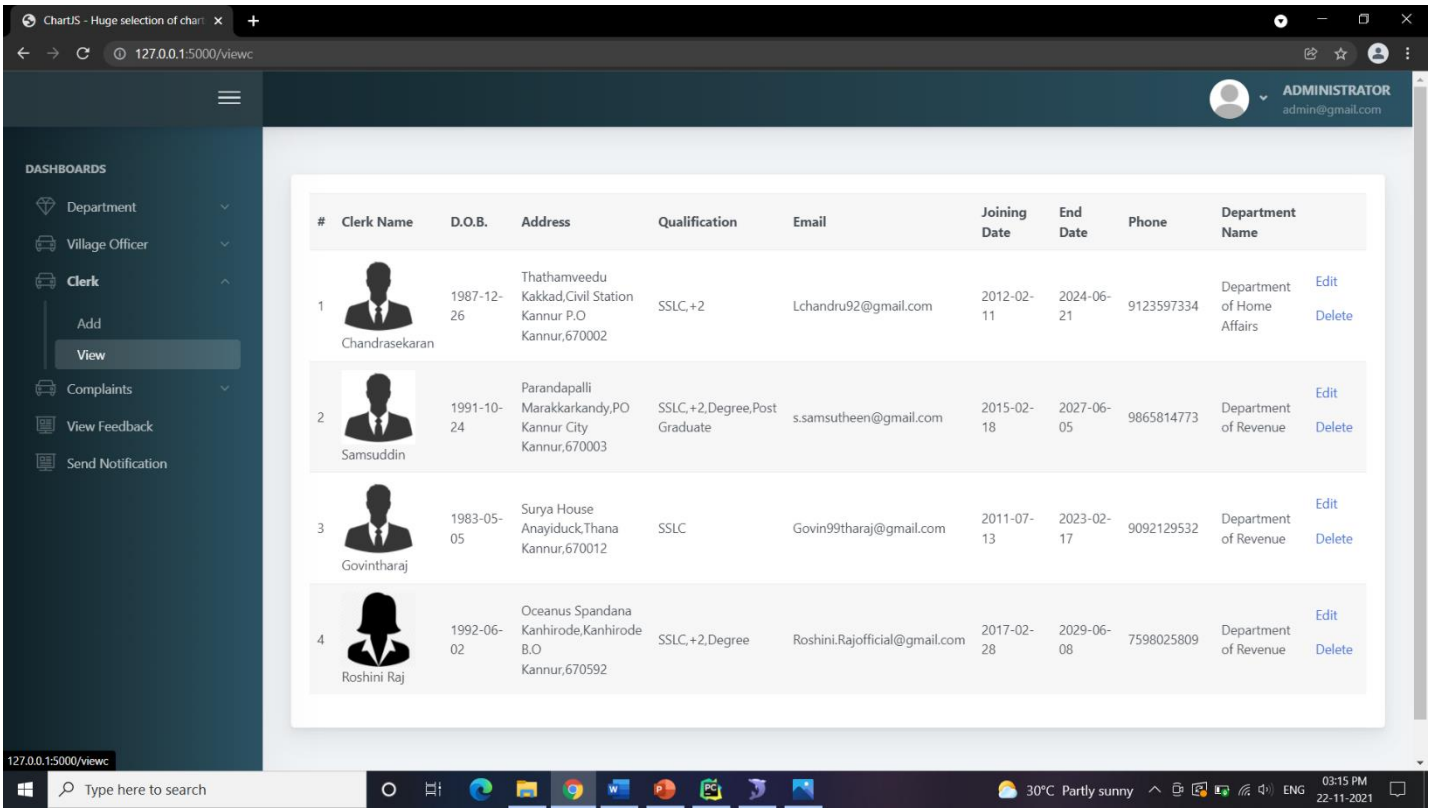
View Village Officer



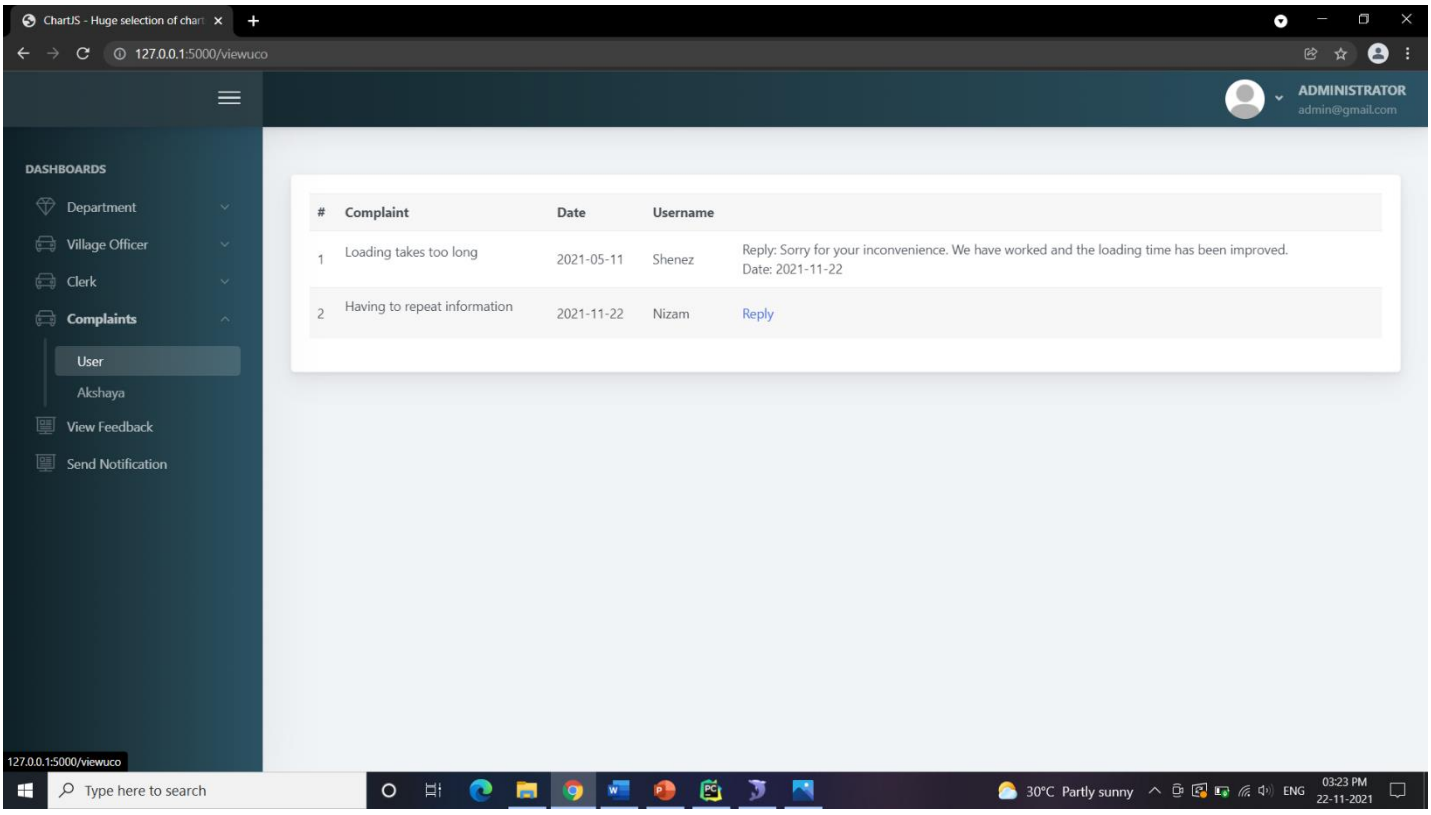
Add Clerk



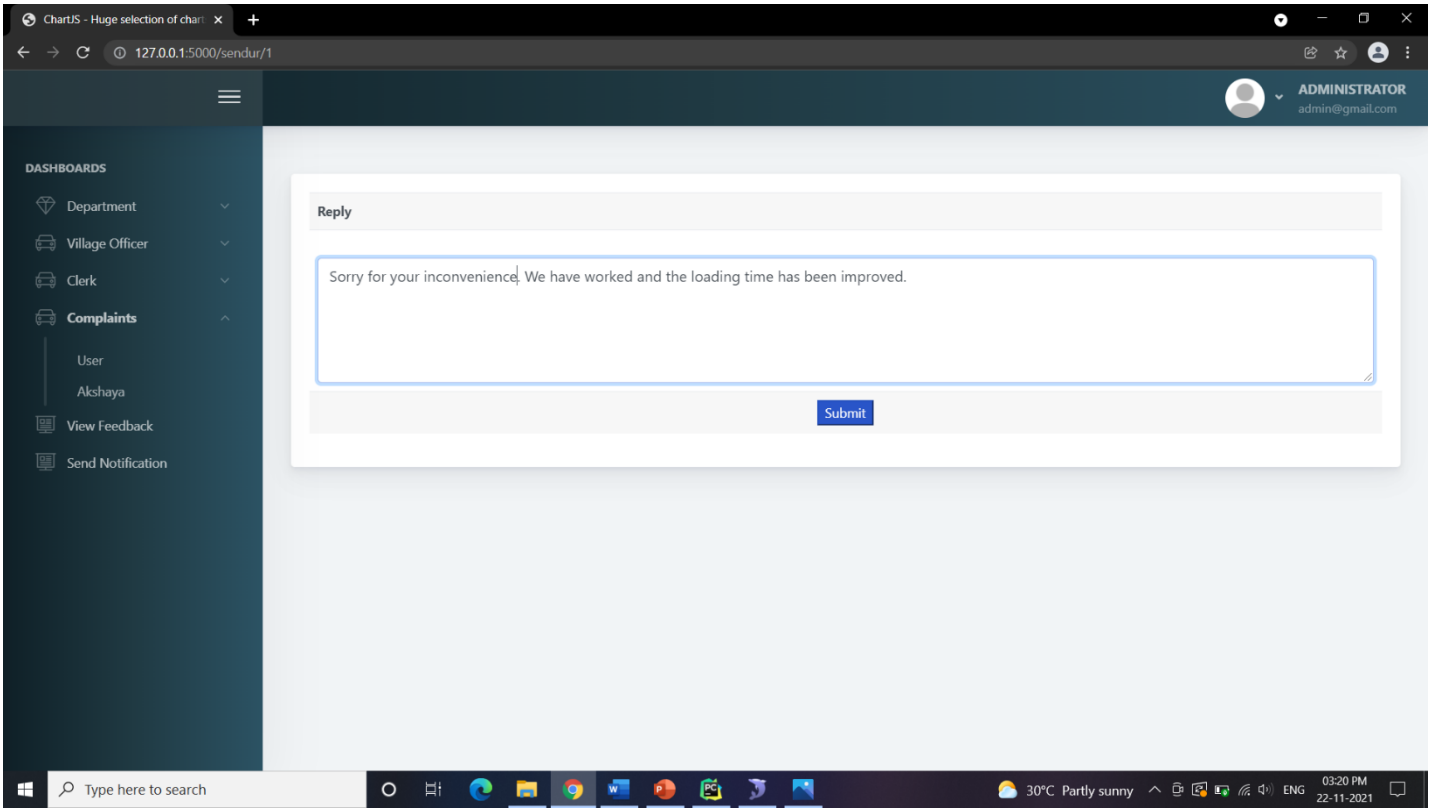
View Clerk



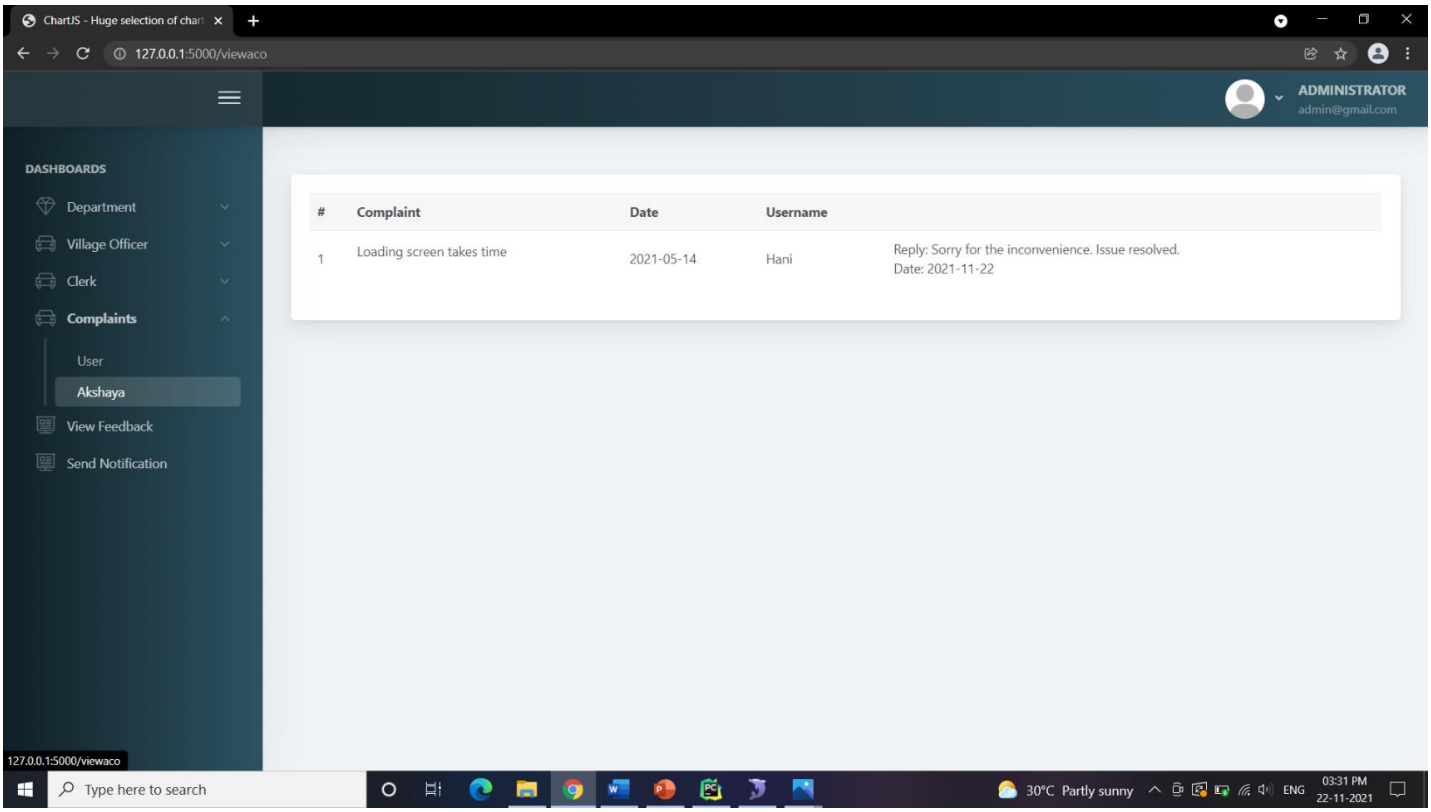
View User Complaints



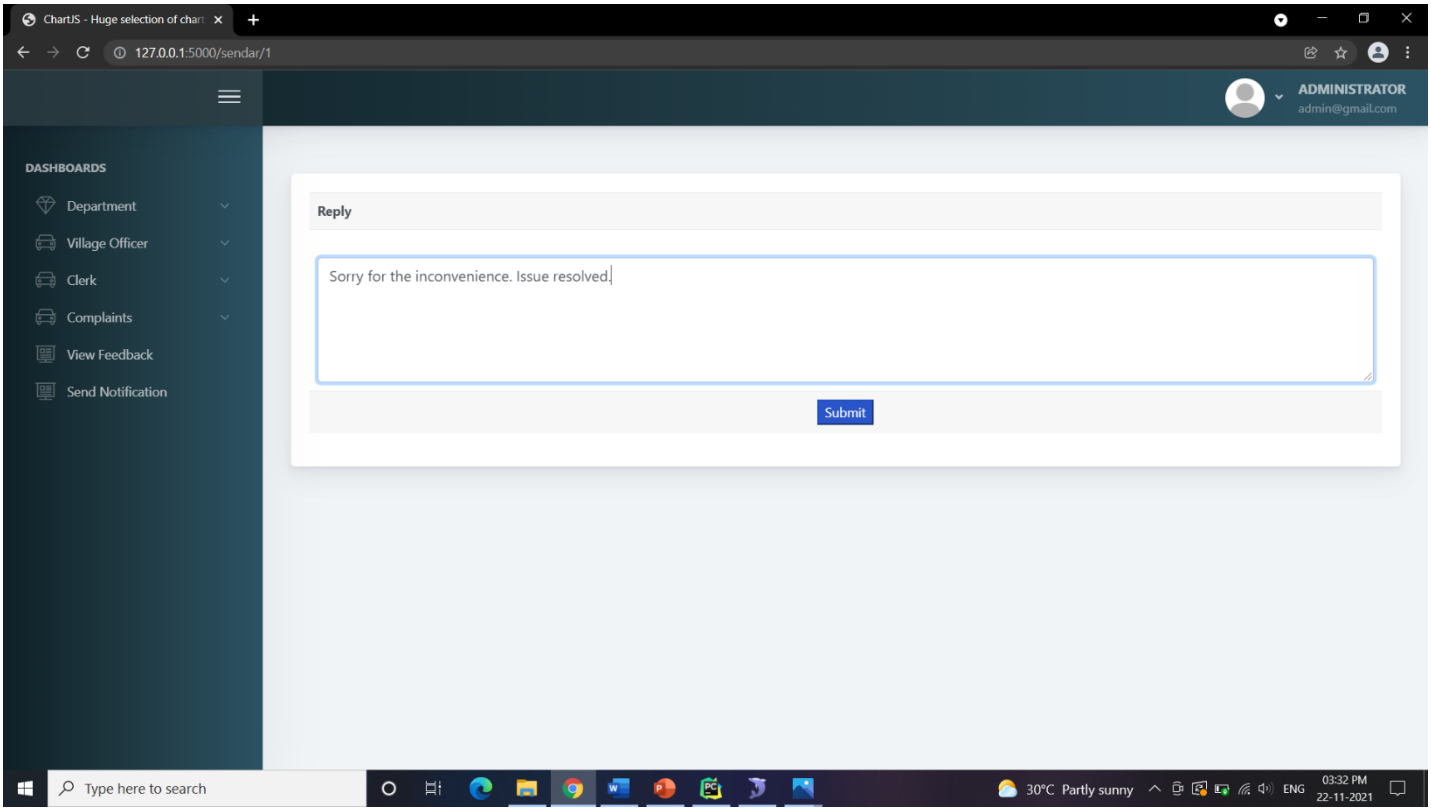
Reply to User Complaint



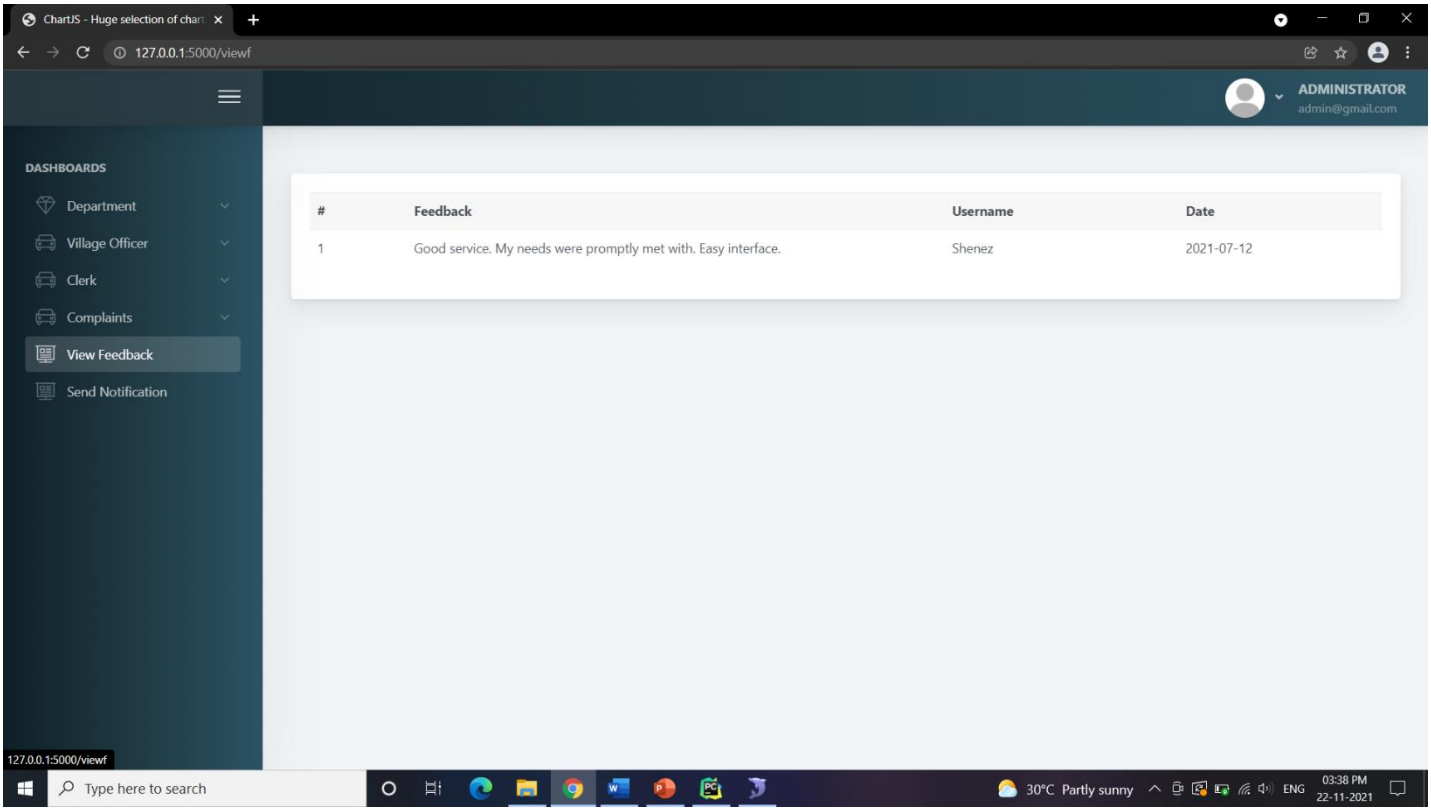
View Akshaya Complaints



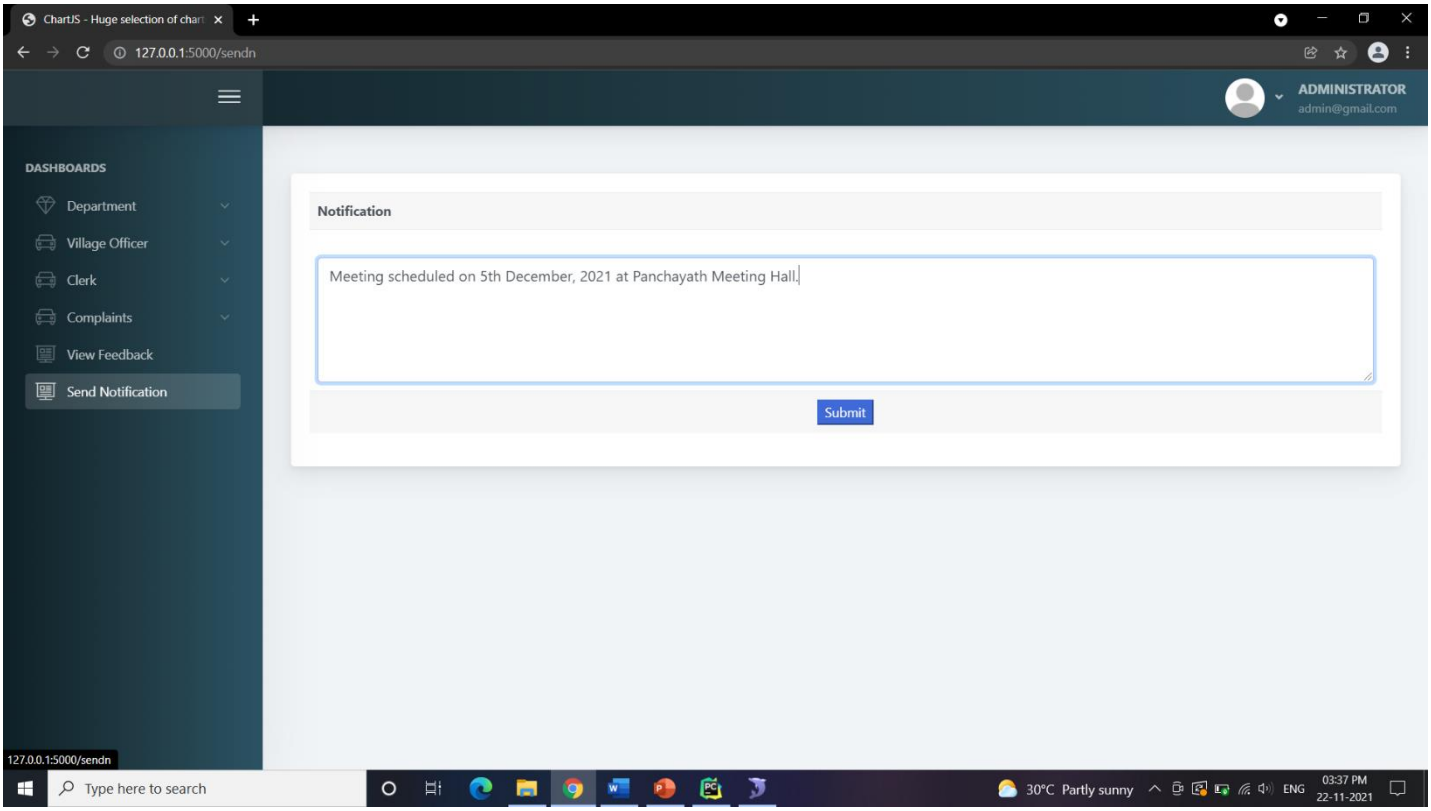
Reply to Akshaya Complaint



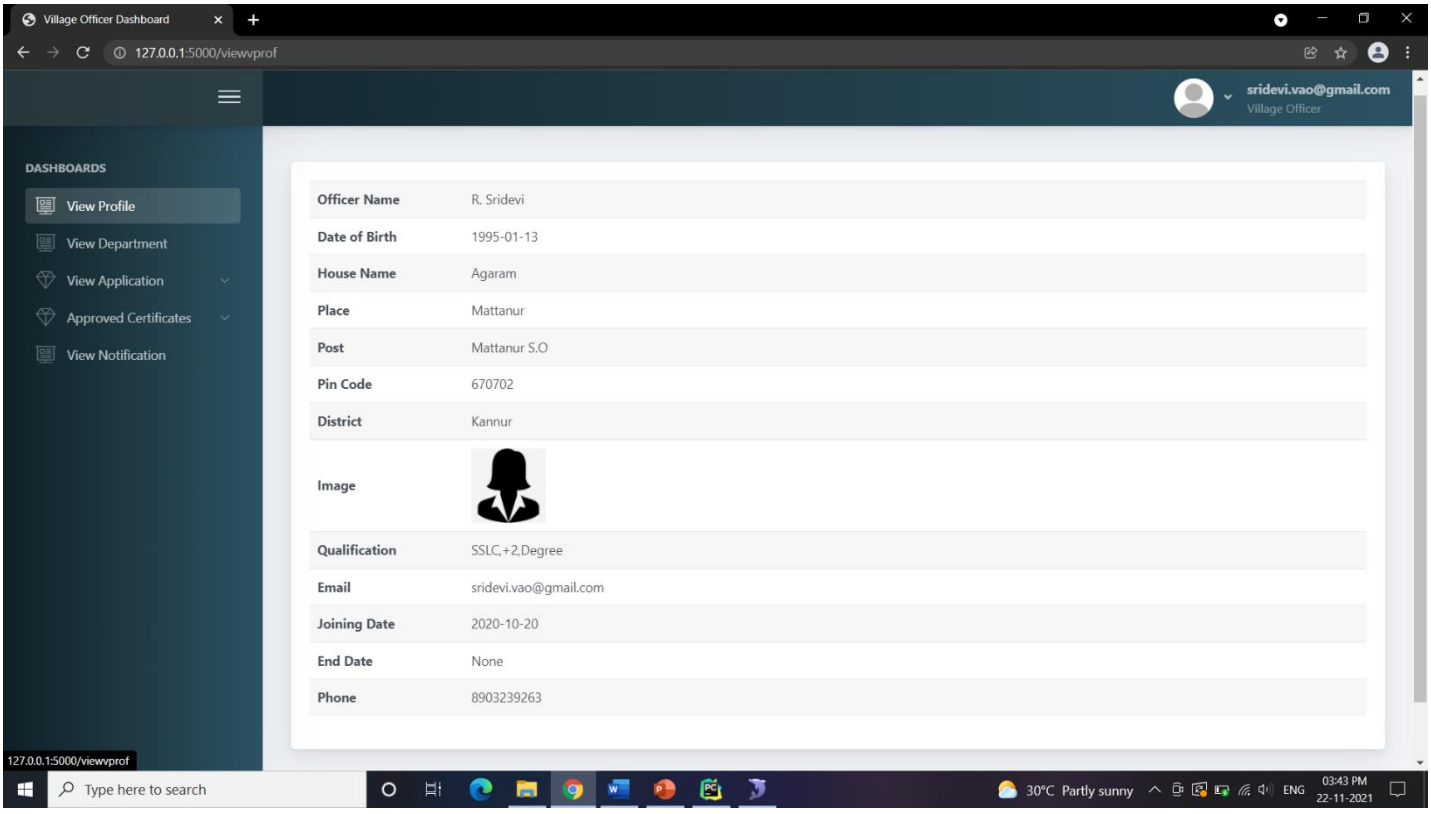
View Feedback



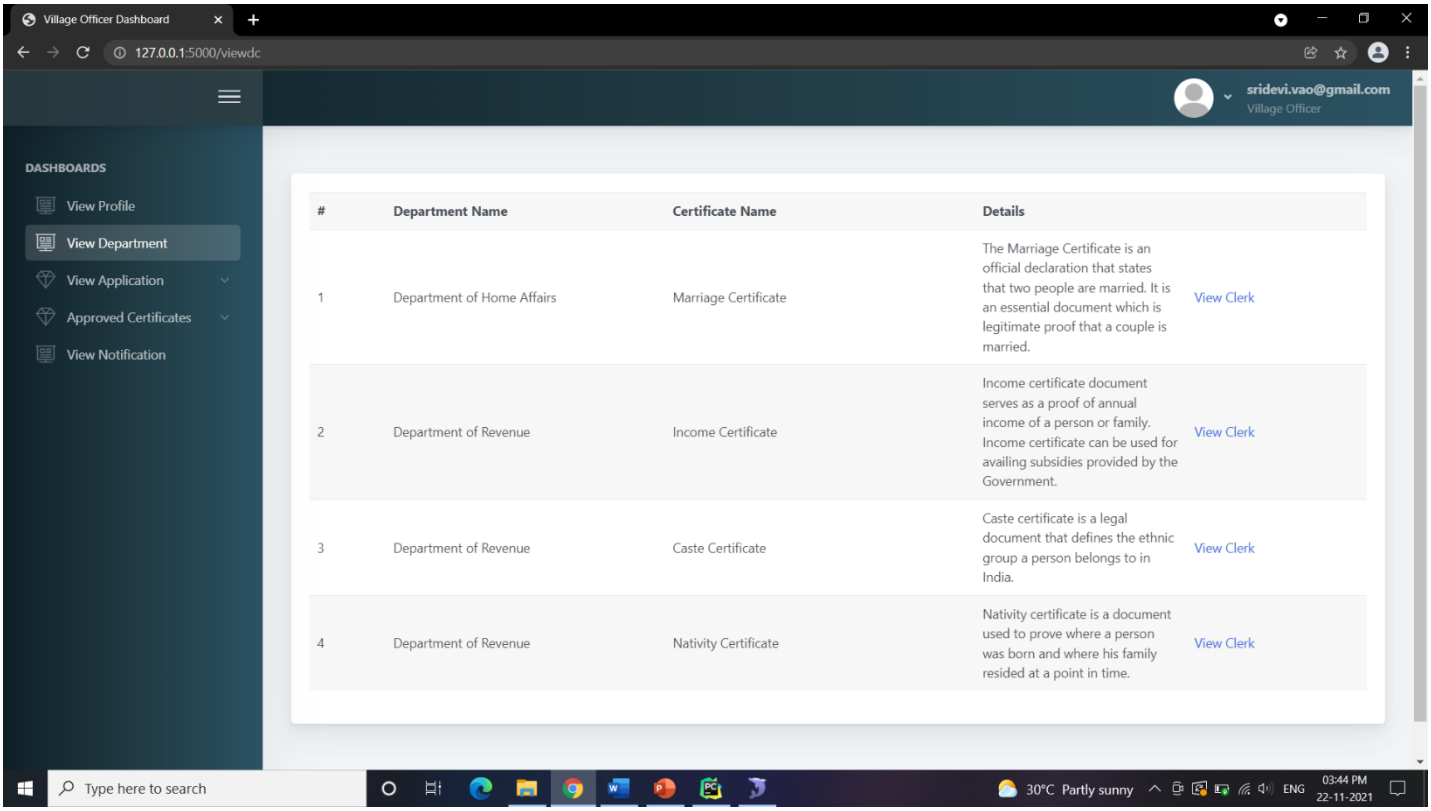
Send Notification

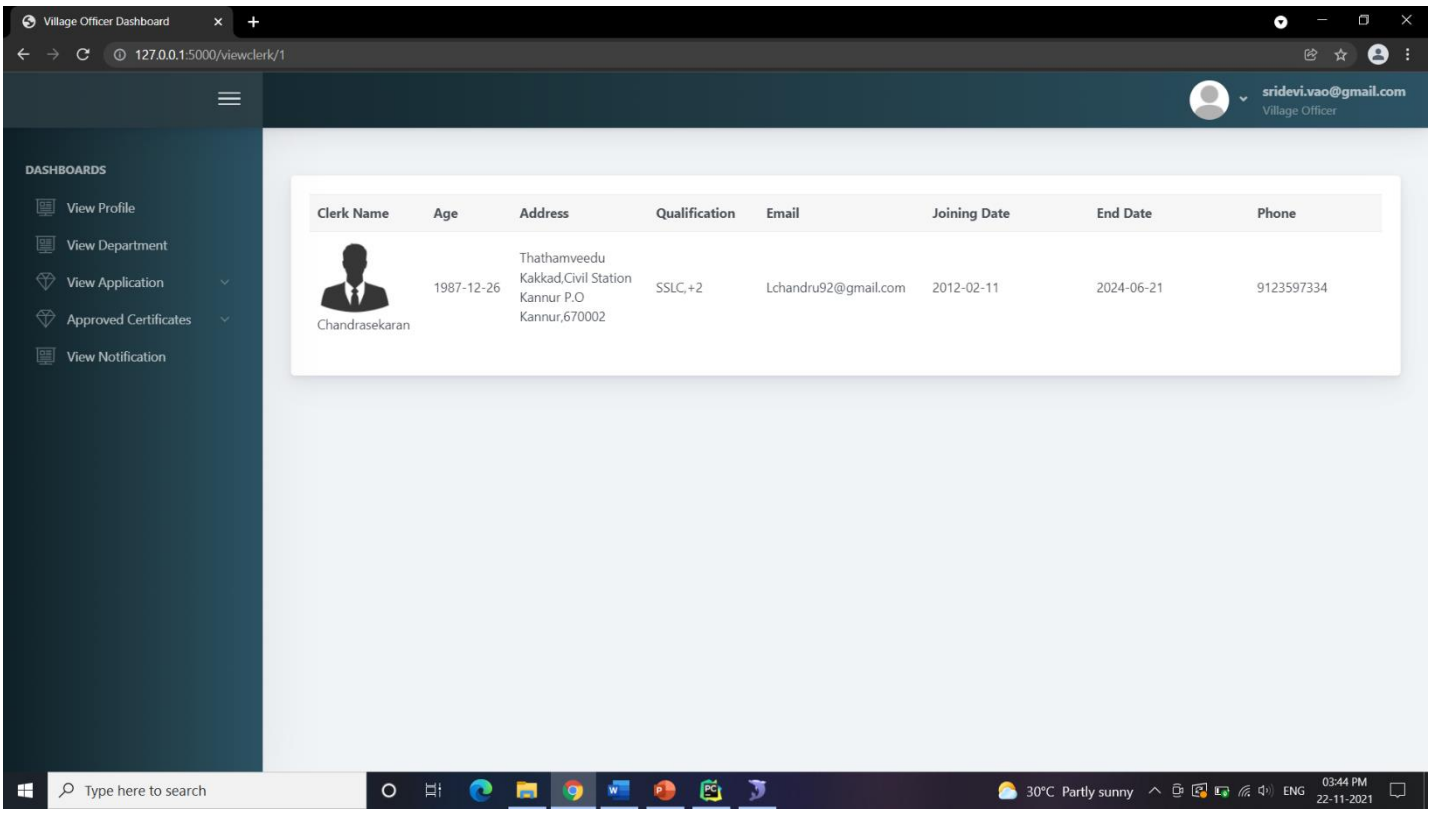


View Village Officer Profile

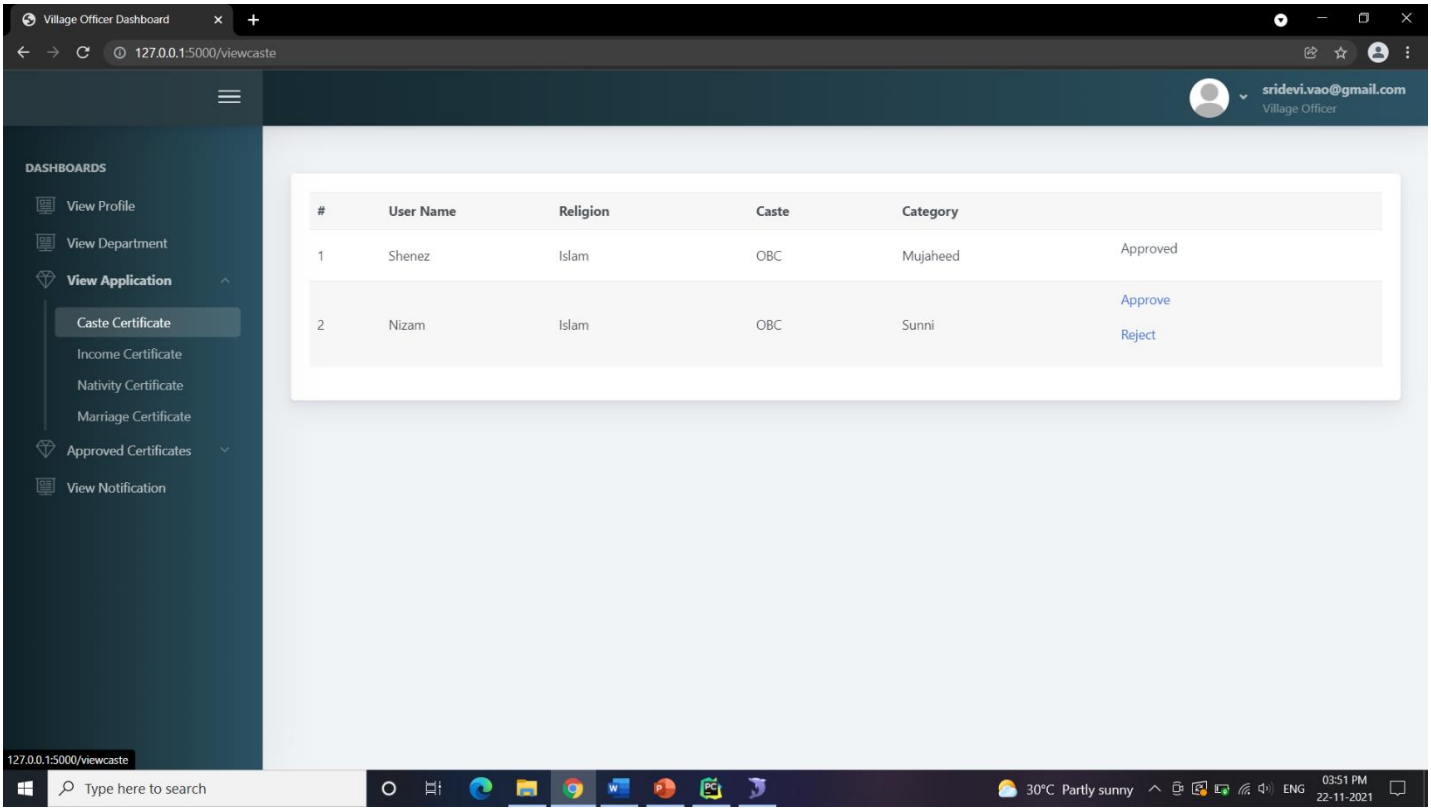


View Department and respective Clerk

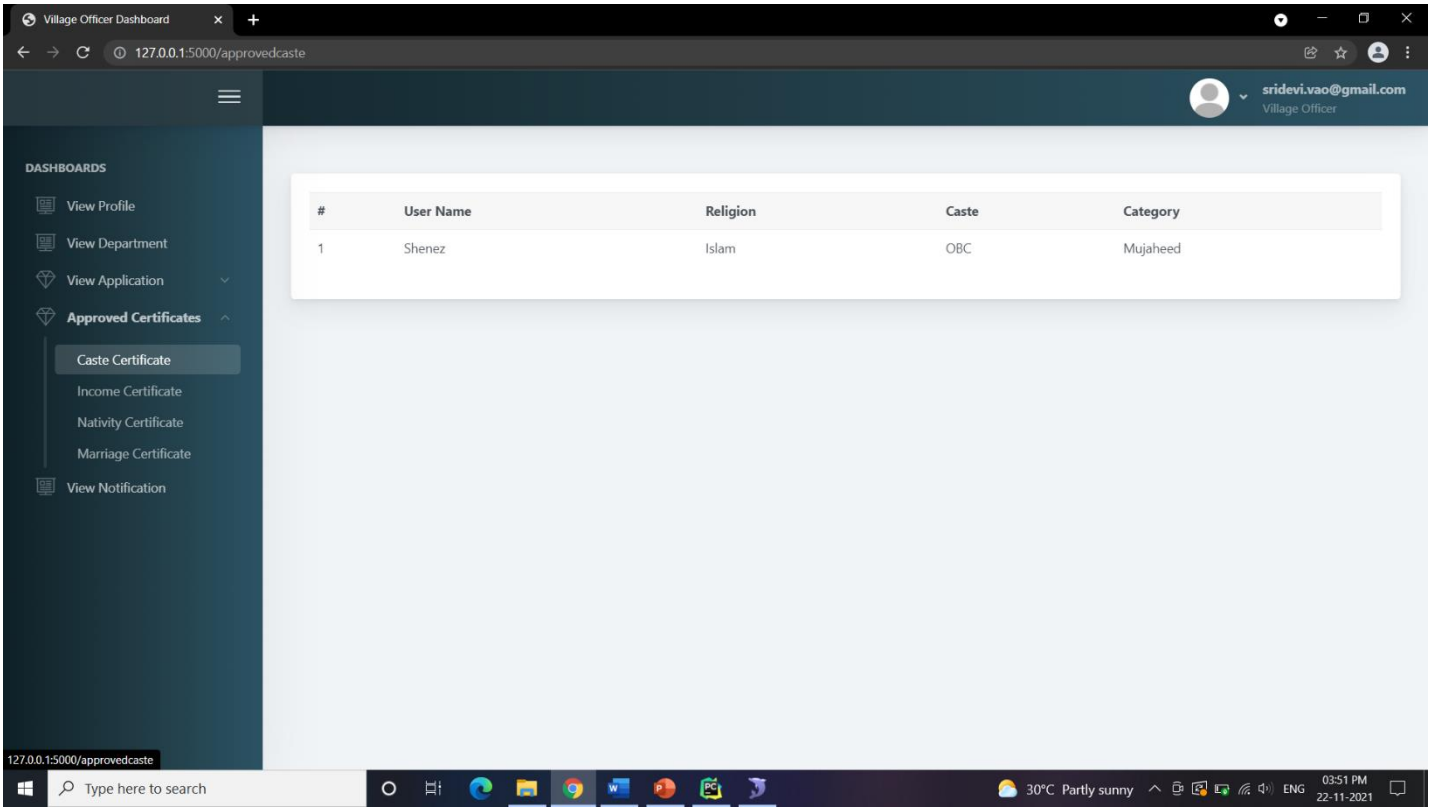




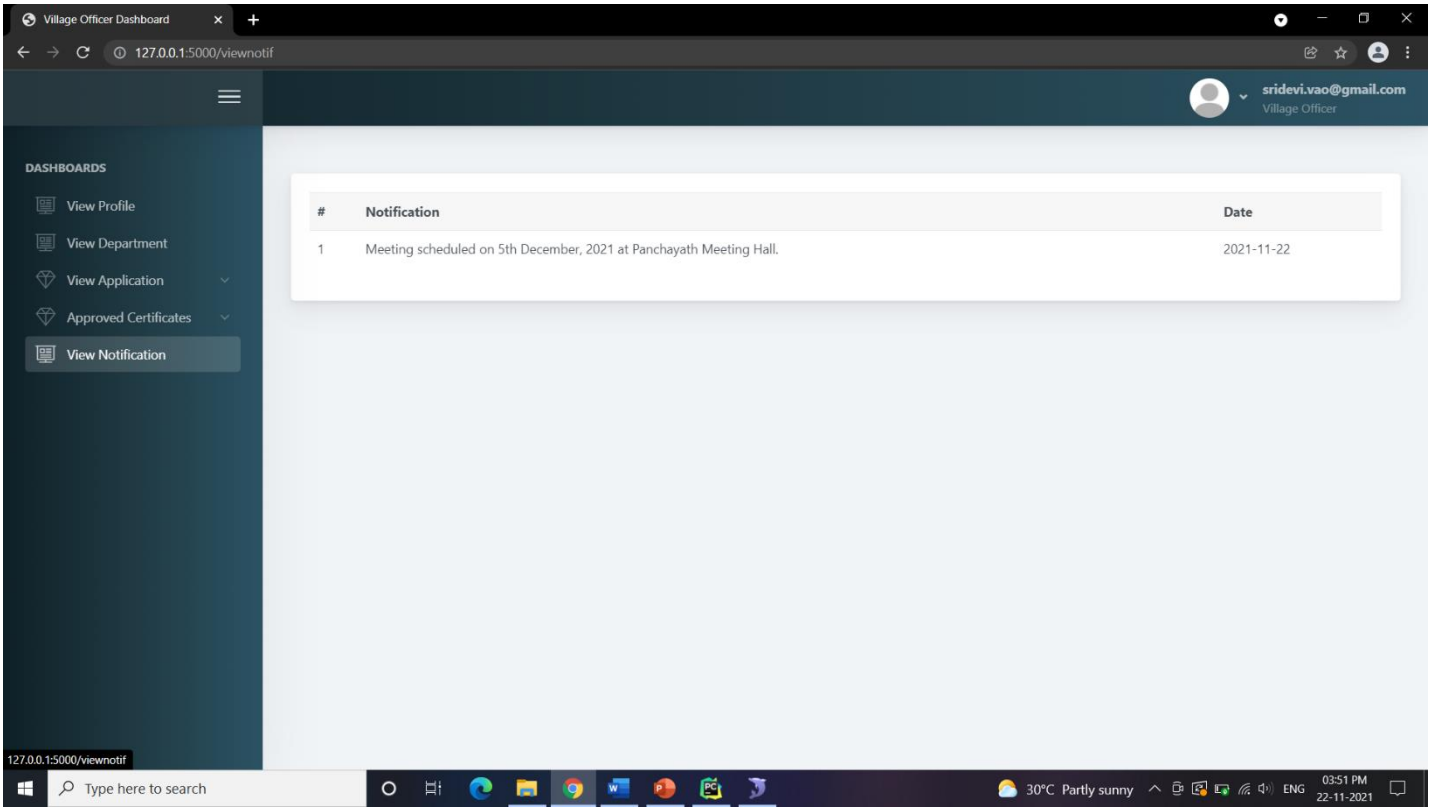
View Application



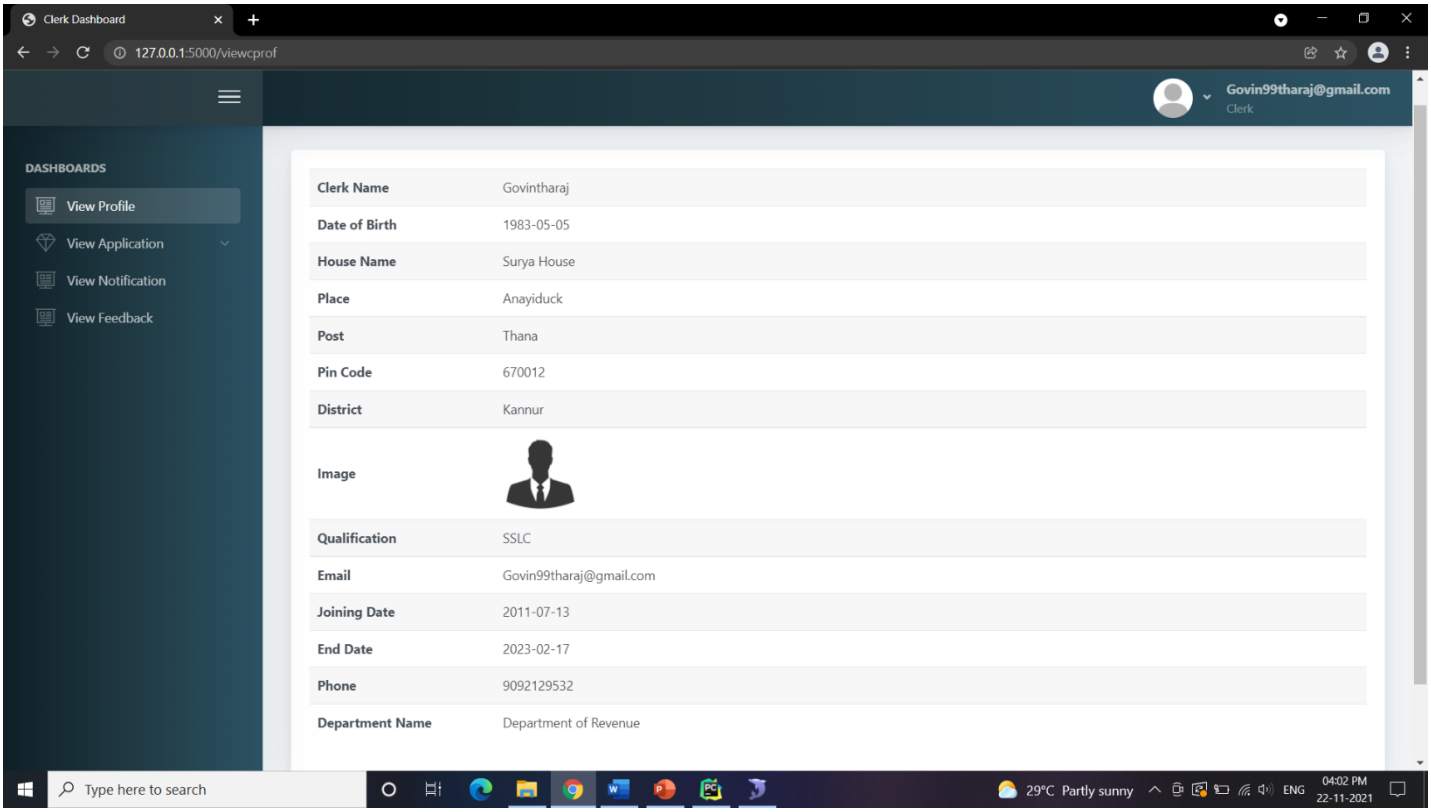
View Approved Certificates



View Notification



View Clerk Profile



View Applications