**Gyan Ganga Institute of Technology and Sciences**

**Jabalpur, Madhya Pradesh**



**Major Project 2 Report**

**Cloud Native Voting Application**

May 2022

|  |  |
| --- | --- |
| Aaryan Soni | 0206CS181001 |
| Adarsh Singh Raghuwanshi | 0206CS181011 |
| Aman Jaisinghani | 0206CS181017 |
| Deepankar Dubey | 0206CS181047 |
| Dev Khare | 0206CS181050 |
| Dwij Jain | 0206CS181056 |

**Certificate**

This is to certify that Aaryan Soni, Adarsh Singh Raghuwanshi, Aman Jaisinghani, Deepankar Dubey, Dev Khare and Dwij Jain, students of class CS-1 have worked on their project entitled “Cloud Native Voting Application” under my supervision. This project is approved for submission towards partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering.

|  |  |
| --- | --- |
| Prof. Sapan Jain | Dr. Ashok Verma |
| Faculty, Computer Science and Engineering | Head of Department,  Computer Science and Engineering |

**Declaration**

We hereby declare that the project entitled “Cloud Native Voting Application” submitted to Gyan Ganga Institute of Science and Technology, Jabalpur, is a record of original work done by our team under the guidance of Dr. Ashok Verma, Head of Department, Computer Science and Engineering.

This project is submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. The results embodied in this thesis have not been submitted to any other University of Institute for the award of any other degree of any degree of diploma.

|  |  |
| --- | --- |
| May 4, 2022 |  |
| Jabalpur |  |

**Acknowledgement**

We are grateful to Dr. Ashok Verma, Head of Department of Computer Science Engineering, Gyan Ganga Institute of Science and Technology, Jabalpur, for providing every help in accomplishing to its final stage the project on “Cloud Native Voting Application” within stipulated time.

The components of this software have been developed in consultation with my associate faculty and we have been greatly benefited by their thoughtful and constructive suggestions, for which we are further thankful to them.

|  |  |
| --- | --- |
| May 4, 2022 | Aaryan Soni  Adarsh Singh Raghuwanshi  Aman Jaisinghani  Deepankar Dubey  Dev Khare  Dwij Jain |
| Jabalpur |

**Table of Content**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 Introduction | | | | | 7 |
|  | | 1.1 Purpose | | | 7 |
|  | | 1.2 Scope | | | 7 |
|  | | 1.3 Target Audience and Assumptions | | | 7 |
|  | | 1.4 Description | | | 8 |
|  | | 1.5 Team Structure and Organisation | | | 8 |
|  | | 1.6 Duration | | | 9 |
| 2 System Overview | | | | | 10 |
|  | | 2.1 Functional Requirements | | | 10 |
|  | | | 2.1.1 Hardware Requirements | | 10 |
|  | | | 2.1.2 Software Requirements | | 10 |
|  | | | 2.1.3 User Interface Requirements | | 11 |
| 2.1.4 Interactive Components | | 11 |
| 2.1.5 Non-Interactive Components | | 11 |
|  | | 2.2 Non-Functional Requirements | | | 12 |
|  | | | 2.2.1 Usability Requirements | | 12 |
| 2.2.2 Security Requirements | | 12 |
| 2.2.3 Performance Requirements | | 13 |
| 2.2.4 Reliability Requirements | | 13 |
| 2.2.5 Project Methodology | | 14 |
|  | 2.3 Design Techniques | | | 15 | |
|  | 2.4 Tier Architecture | | | 18 | |
| 3 Diagrams | | | | | 19 |
| 4 Screenshot | | | | | 22 |
|  | 4.1 Code Snippets | | | 24 | |
| 5 Software Testing | | | | | 26 |
| 6 Business Diagram | | | | | 28 |
| 7 Future Prospects | | | | | 29 |
| 8 Conclusion | | | | | 30 |
| 9 References | | | | | 32 |

1. **Introduction**
   1. Purpose

An online voting app is a Software Application that allows groups to securely conduct votes and elections.

High quality online voting applications provide security, accessibility, and an overall requirement to conduct the body in an efficient way.

* 1. Scope

At their core, online voting applications protect the integrity of your vote by preventing voters from being able to vote multiple times. As a digital platform, they eliminate the need to gather in-person, cast votes using paper, or by any other means (e.g. email, insecure survey software).

You may hear an online voting application being referred to as an online election system, an online e voting system, or electronic voting. These all refer to the same thing: a secure voting tool that allows your group to collect input from your group and closely scrutinise the results in real time.

1.3 Target Audience and Assumption  
Anyone with some programming experience can understand this document.

The document is intended for developers, software architects, testers, project managers and documentation writers.

1.4 Description

Cloud Native Voting Application is be a desktop software application in order to provide all the required features mentioned in the document.

Product Features: This software package will allow organizations to design and run polls securely and privately on a dedicated server. It will be possible to customize the polls before every round of voting. The software will provide voters to register their votes on the different alternatives that are available and helps them view the results alongside it. The result data from the voters will be used to calculate statistics and displays the percentage share of all the alternates provided.

1.5 Team Structure and Organisation

|  |  |  |
| --- | --- | --- |
| **Name** | **Roles** | |
| Aaryan Soni | Backend | Networking |
| Adarsh Singh Raghuwanshi | Oracle Cloud |
| Dwij Jain | Containerization |
| Aman Jaisinghani | Web Dev | HTML, CSS, JS |
| Deepankar Dubey | Flask, Angular |
| Dev Khare | Documentation and Testing | |

**Team 1**

The task of team 1 is to make the application cloud native.

**Team 2**

The assigned task of team 2 was to develop the website.

**Team 3**

**Documentation and Testing**: The responsibility of Dev Khare was to keep documenting the progress made by the team, handle version control, and continuously integrate progress made after testing code.

1.6 Duration

|  |  |  |  |
| --- | --- | --- | --- |
| **SNo.** | **Project Phase** | | **Working Days** |
| 1 | Business Requirements | | 7 |
| 2 | Application Design | | 7 |
| 3 | Development and Testing | | 20 |
| 4 | Documentation | | 7 |
| 5 | Deployment User Acceptance Testing | | 4 |
| 6 | Project Management | | 5 |
|  | | Total | 50 |
|  | | |  |

1. **System Overview**

This section is an overview of the main features of the project. Requirements are structured by functionality.

* 1. Functional Requirements
* Internet
* List of candidates eligible for voting (candidate list).
* Web application - A modern browser (Chrome, Firefox, Opera).
* Eligible parties for which a vote can be casted.
* A national identification card (Voter Id, Aadhar Card).  
  + 1. Hardware Requirements  
         
       PC with minimum Pentium Dual Core.  
       RAM is more than 2 GB.  
       Keyboard and mouse or any equivalent device.
    2. Software Requirements  
         
       Google Chrome or any latest browser.  
       OS: Linux or Windows with WSL.  
       Latest version of Docker installed on server.
    3. User Interface Requirements

User interface requirements are briefly mentioned below:

1. Content presentation

2. Easy Navigation

3. Simple interface

4. Responsive

5. Consistent UI elements

6. Default settings

7. User centric approach

* + 1. Interactive Components  
         
       There are two sections of the website: vote and view result.  
       Voting: User can select a candidate to vote for.  
       View result: Result is displayed for the user.

* + 1. Non-interactive Components

Non-interactive components of this project are icons, background colours, button shapes and sizes, and other static components required to make the application work smoothly.

* 1. Non-functional Requirements

This section describes the non-functional requirements of the project. The non-functional requirements are to secure against attackers, the application should be easy to use, automatic, universally available and for individuals.

* + 1. Usability Requirements

The application has a simple interface which is intuitive to use for the user. Though it does not have the latest design guidelines, it still provides all the basic functionality for user interaction. The system developed is intended to be used by the general public and should not have a steep learning curve. Also, anyone around the globe can access the application with no barriers.

* + 1. Security Requirements

Anonymous voting  
Data transfer is secure

* + 1. Performance Requirements  
         
       Vote should be casted in a timely manner.  
       Casting should be guaranteed.  
       The entire transaction takes place at once or doesn't happen at all.  
       The database must be consistent before and after the transaction.  
       Multiple transactions occur independently without interference.  
       The changes of a successful transaction occur even if the system failure occurs.

2.2.4 Reliability Requirements

Reliability is an attribute which specifies how likely the system or its element would run without a failure for a given period of time under predefined conditions.

2.2.5 Project Methodology

Agile and Scrum software development approach was used to develop this project. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages flexible responses to change. Scrum is one of the methodologies that follow Agile principles. It is a combination of iterative and incremental models. Agile focuses on a single task with daily collaboration and communication and focuses on fast delivery.

Agile software development model, when compared to traditional software development models is fast, flexible and cost effective. By clarifying an acceptance criterion, estimation and breakdown of user stories, tasks can be broken down that can fit in a spirit cycle.

In the spirit pre planning phases of this project, user stories were estimated and broken down into Themes, Epics, Stories and Tasks. A task was assigned to each team member. The tasks when completed were reviewed, tested and then used as a base for the next spirit cycle.

Agile software development approach enabled us to have small scale focuses with rapid development cycles and minimised resources allocated to testing.

2.3 Design Techniques

**Cloud**

"The cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centres all over the world. By using cloud computing, users and companies do not have to manage physical servers themselves or run software applications on their own machines.

The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data centre, instead of locally on the user device. This is why a user can log in to their Instagram account on a new phone after their old phone breaks and still find their old account in place, with all their photos, videos, and conversation history.

https://www.youtube.com/watch?v=dQw4w9WgXcQ

Made in Discord Adda tapra-1

**Java**

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets.

We have used the following properties of Java:

• Variable Declaration: The types of all variables must be declared. The primitive types are byte, short, int, long (8, 16, 32, and 64 bit integer variables, respectively), float and double (32 and 64-bit floating point variables), Boolean (true or false), and char. Boolean is a distinct type rather than just another way of using integers. Strings are not a primitive type, but are instances of the String class. Because they are so common, string literals may appear in quotes just as in other languages.

• Classes are effectively new programmer-defined types; each class defines data (fields) and methods to manipulate the data. Fields in the class are template for the instance variables that are created when objects are instantiated (created) from that class. A new set of instance variables is created each time that an object is instantiated from the class.

**Redis**

Redis is an open source (BSD licensed), in-memory data structure store used as a database, cache, message broker, and streaming engine. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions, and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

**PostgreSQL**

PostgreSQL, commonly pronounced “Post-GRES,” is an open source database that has a strong reputation for its reliability, flexibility, and support of open technical standards. Unlike other RDMBS (Relational Database Management Systems), PostgreSQL (link resides outside ibm.com) supports both non-relational and relational data types. This makes it one of the most compliant, stable, and mature relational databases available today.

2.4 Tier Architecture

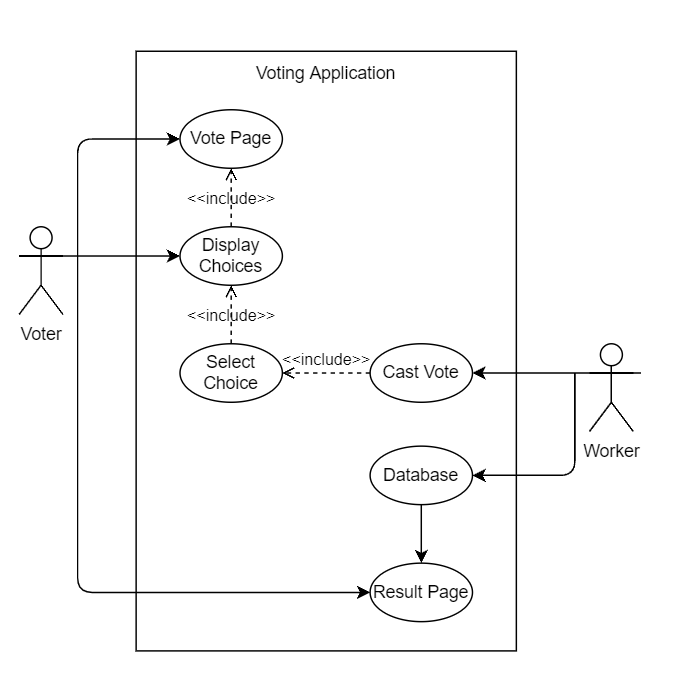
**Presentation Layer** - The web site or the voting application is called the presentation layer. The presentation layer is the most important layer simply because it’s the one that everyone sees and uses. Even with a well-structured business and data layer, if the presentation layer is designed poorly, this gives the users a poor view of the system. Presentation layer is the form where we design using the controls like textbox, labels, command buttons etc.

**Business Layer**: Though a web site could talk to the data access layer directly, it usually goes through another layer called the business layer. This layer is a class which we use to write the function which works as a mediator to transfer the data from Application or presentation layer data layer. In the three-tier architecture we never let the data access layer to interact with the presentation layer.

**Data layer**: The key component to most applications is the data. The data has to be served to the presentation layer somehow. The data layer is a separate component whose sole purpose is to serve up the data from the database and return it to the caller.

1. **Diagrams**

Use Case Diagram



|  |
| --- |
| Sequence Diagram    Data Flow Diagram Level 0    Data Flow Diagram Level 1 |

|  |
| --- |
| Entity Relationship |
|  |
|  |

**4. Screenshots**

|  |  |  |
| --- | --- | --- |
|  | | Voting Page |
|  | |  |
| Vote Selected |  | |
|  |  | |
|  | | Result Page |
|  | |  |
| Result Page after Votes |  | |

|  |  |  |
| --- | --- | --- |
|  | | Buildtime Logs |
|  | |  |
| Runtime Logs |  | |
|  |  | |
|  | | Docker Images |
|  | |  |
| Docker Containers |  | |

4.1 Code Snippet

**Frontend**

|  |
| --- |
| <body>  <div id=**"content-container"**>  <div id=**"content-container-center"**>  <form id=**"choice"** name=**'form'** method=**"POST"** action=**"/"**>  <button id=**"a"** type=**"submit"** name=**"vote"** class=**"a"** value=**"a"**>**{{option\_a}}**</button>  <button id=**"b"** type=**"submit"** name=**"vote"** class=**"b"** value=**"b"**>**{{option\_b}}**</button>  </form>  </div>  </div>  <script src=**"http://code.jquery.com/jquery-latest.min.js"** type=**"text/javascript"**></script>  <script src=**"//cdnjs.cloudflare.com/ajax/libs/jquery-cookie/1.4.1/jquery.cookie.js"**></script>  </body> |

**Backend**

|  |
| --- |
| package worker**;**  **import** redis**.**clients**.**jedis**.**Jedis**;**  **import** redis**.**clients**.**jedis**.**exceptions**.**JedisConnectionException**;**  **import** java**.**sql**.\*;**  **import** org**.**json**.**JSONObject**;**  class Worker **{**  public static void main**(**String**[]** args**)** **{**  **try** **{**  Jedis redis **=** connectToRedis**(**"redis"**);**  Connection dbConn **=** connectToDB**(**"db"**);**  System**.**err**.**println**(**"Watching vote queue"**);**  **while** **(true)** **{**  String voteJSON **=** redis**.**blpop**(**0**,** "votes"**).**get**(**1**);**  JSONObject voteData **=** **new** JSONObject**(**voteJSON**);**  String voterID **=** voteData**.**getString**(**"voter\_id"**);**  String vote **=** voteData**.**getString**(**"vote"**);**  System**.**err**.**printf**(**"Processing vote for '%s' by '%s'\n"**,** vote**,** voterID**);**  updateVote**(**dbConn**,** voterID**,** vote**);**  **}**  **}** **catch** **(**SQLException e**)** **{**  e**.**printStackTrace**();**  System**.**exit**(**1**);**  **}**  **}**  static void updateVote**(**Connection dbConn**,** String voterID**,** String vote**)** **throws** SQLException  **{** **...** **}**  static Jedis connectToRedis**(**String host**)**  **{** **...** **}**  static Connection connectToDB**(**String host**)** **throws** SQLException  **{** **...** **}**  static void sleep**(**long duration**)**  **{** **...** **}**  **}** |

**5 Software Testing**

Testing is the process of executing a program with the aim of finding errors. We perform testing of our system before deployment for efficient functioning of our system.

Types of Testing:

1. Functional Testing  
   It ensures that the application is functioning correctly. This type of testing focuses on the main purpose and flow of the application, ensuring that all its features are responsible and meet specifications
2. Usability Testing  
   Usability testing is a technique used in user-centred interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system. It is more concerned with the design intuitiveness of the product and tested with users who have no prior exposure to it. Such testing is paramount to the success of an end product as a fully functioning application that creates confusion amongst its users will not last for long.
3. Compatibility Testing  
   Compatibility testing is a part of non-functional testing conducted on application software to ensure the application's compatibility with different computing environments.

**Web Interface Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Scenario** | **Expected Result** | **Result** |
| User | Visits URL | Website loads | Success |
| Vote Select | User selects candidate | Data entered | Success |
| Vote Cast | User votes | Vote is cast | Success |
| Result View | User views result | Result is displayed | Success |

**Application Testing**

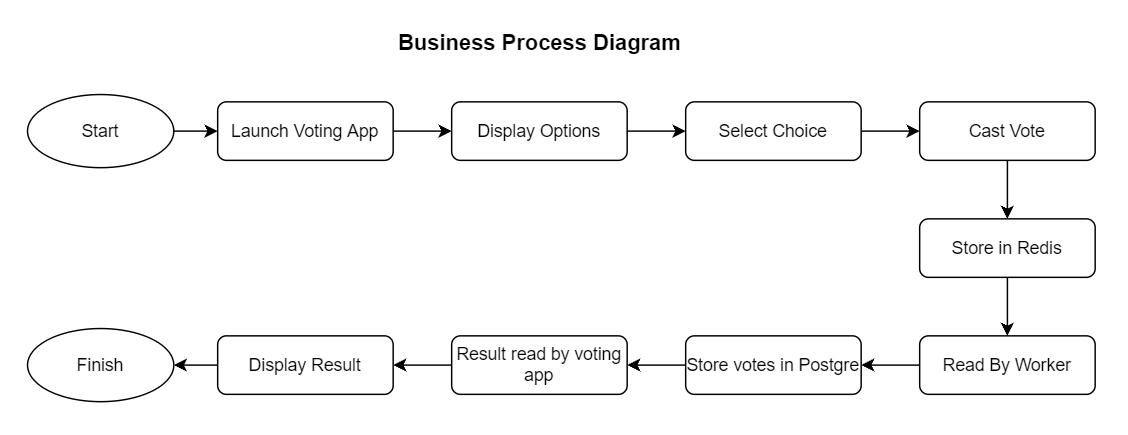
|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Scenario** | **Expected Result** | **Result** |
| Load Testing | High traffic | Website handles load | Success |
| Cache Database | Data is fed | Details are recorded | Success |
| Worker | Fetches data from cache database | Data is fetched | Success |
| Persistent Database | Data is fed by Worker | Data is recorded | Success |

**6 Business Model**

**Business Requirement**Business Requirement focuses on the business perspective as it holds the details of the business solution for a project. It defines what the organization wants or needs to be able to do once the project is completed and also describes the changes in capabilities that will result from the project.

Business Process Modelling Notation (BPMN) is a flow chart method that models the steps of a planned business process from end to end.

Its purpose is to model ways to improve efficiency, account for new circumstances or gain competitive advantage. The method has been undergoing a standardisation push in the past few years and is now often called by a slightly different name: **Business Process Model and Notation**, still using the BPMN acronym. It differs from Unified Modelling Language (UML) used in software design.



1. **Future Prospects**

The Online Voting System platform can be made more secure by using the following methods:

1. Password Changing
2. Fingerprinting
3. Cornea Detection

The password used by the user to vote is provided by the administrator. In the future the user can be given the privilege of changing the password. So it helps to increase the security of the system.

The other two methods that can be used are cornea detection and fingerprinting. But here the problem is that it decreases the scope of the platform because these systems need some electronic components to implement. So it will avoid the users privilege to cast the votes at their fingertips. But it can guarantee that fake voting will be impossible.

**8 Conclusion**

Using an online voting tool will generate confidence in the results of your votes and elections, lower your voting-related costs, and streamline the election process for both you and your voters.

Cost Savings and Efficiency

The cost savings and efficiencies you’ll gain are unparalleled to any other method of voting. Groups switching to web-based online voting systems from more expensive and less efficient voting technologies like voting machines, paper ballots, and in-person meetings will reap these benefits without increasing risk.

Voter Accessibility

Needing to fly halfway around the world to vote at your organisation’s annual meeting is an example of a vote with low accessibility. On the other hand, tapping a link on your mobile device that securely logs you into the online voting system website is an example of a vote or election with high accessibility.

High accessibility generates greater turnout rates among your group.

Auditability and Verifiability

With an online voting system, you can easily showcase election results to eliminate concern. Sharing all administrator activity during your election to prove no one went in altered the results is just one of the many trust-building tactics you’ll be able to use in light of a vote challenge.

Security, Confidence, and Trust in Your Election Results

The confidence in your voting and election results is by far the most valuable aspect that online voting systems will offer to your group. The fallout of a vote being perceived as unfair is expensive, time-consuming, and wreaks havoc on the hard-earned trust you’ve built among group members. From this perspective, an online voting system offers unparalleled election security.

**9 References**

1 AWS

<https://docs.aws.amazon.com/general/latest/gr/Welcome.html>

<https://docs.aws.amazon.com/ec2/index.html>

2 Docker

<https://docs.docker.com/reference/>

3 PostgreSQL

<https://www.postgresql.org/docs/current/>

4 Redis  
<https://redis.io/commands>