SAVITRIBAI PHULE PUNE UNIVERSITY

A PROJECT REPORT ON

# **Devnagari CAPTCHA Generation**

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

**BACHELOR OF ENGINEERING**

**(Computer Engineering)**

**SUBMITTED BY**

**BHAVANA THAKARE Exam No: 72018608M**

**MRUNAL SADAWARTE Exam No: 72018470D**

**SAKSHI GHONGE Exam No: 72018080F**

**ABHIRAM SHELKAR Exam No: 72017869L**

Under The Guidance of

**Prof. Mrs. S. T. Somvanshi**

**Prof. Mrs. Yasmin Khan**



**DEPARTMENT OF COMPUTER ENGINEERING**

Dr. D. Y. Patil College of Engineering Akurdi

College Address

# **CERTIFICATE**

This is to certify that the Project Entitled

# **Devnagari CAPTCHA Generation**

Submitted by

**BHAVANA THAKARE Exam No: 72018608M**

**MRUNAL SADAWARTE Exam No: 72018470D**

**SAKSHI GHONGE Exam No: 72018080F**

**ABHIRAM SHELKAR Exam No: 72017869L**

Is a bonafide work carried out by Students under the supervision of **Prof. Mrs. S. T.** **Somvanshi** and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering).

|  |  |
| --- | --- |
| Prof. Mrs. S. T. Somvanshi  Prof Mrs. Yasmin Khan | Dr . Prof. M.A. Pote |
| Internal Guide, | H.O.D |
| Dept. of Computer Engg | Dept. of Computer Engg. |
|  |  |
|  |  |

Principal

D. Y. Patil College of Engineering, Akurdi Pune 44

Signature of Internal Signature of External

Examiner Examiner

**PROJECT APPROVAL SHEET**

A Project Title

# **Devnagari CAPTCHA Generation**

Is successfully completed by

BHAVANA THAKARE Exam No: 72018608M

MRUNAL SADAWARTE Exam No: 72018470D

SAKSHI GHONGE Exam No: 72018080F

ABHIRAM SHELKAR Exam No: 72017869L

At

**DEPARTMENT OF COMPUTER ENGINEERING**

**Dr. D. Y. Patil College of Engineering Akurdi**

**SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE**

ACADEMIC YEAR 2022-2023

|  |  |
| --- | --- |
| Prof. Mrs. S. T. Somvanshi  Prof Mrs. Yasmin Khan | Dr . Prof. M.A. Pote |
| Internal Guide, | H.O.D |
| Dept. of Computer Engg | Dept. of Computer Engg. |

**ABSTRACT**

This report describes an alternative methodology for the generation of CAPTCHA which can be used as a language barrier for people who can only understand their native language.

The proposed methodology involves a system of text-based CAPTCHA that is based on Devanagari script which is written in the form of Indian languages that is Hindi and Marathi.

This report discusses the process of creating abstract texts that use the Devanagari script, which is an essential part of many Asian languages such as Hindi, Marathi, Nepali, etc. This report highlights the potential benefits of adding a Devanagari language in the generation of CAPTCHAs.

It is implemented for the better understanding of Indian people. This system will allow people to understand and use captchas in rural parts. Users may use this in any form of login resulting in the swift working of the process.

**Acknowledgment**

# *It gives us great pleasure in presenting the preliminary project report on “***Devnagari CAPTCHA Generation”**.

*I would like to take this opportunity to thank my internal guide* ***Prof. Mrs. S. T. Somvanshi*** *for giving me all the help and guidance I needed. I am grateful to them for their kind support. Their valuable suggestions were very helpful.*

*I am also grateful to Prof. HOD Name, Head of Computer Engineering Department, Dr. D. Y. Patil College Akurdi for his indispensable support, suggestions.*

*In the end our special thanks to Other Person Name for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.*

**NAME OF THE STUDENTS**

**BHAVANA THAKARE**

**MRUNAL SADAWARTE**

**SAKSHI GHONGE**

**ABHIRAM SHELKAR**

# Contents

1. **Synopsis**  1
   1. Project Title 1
   2. Project Option 1
   3. Internal Guide 1
   4. Technical Keywords (As per ACM Keywords) 1
   5. Problem Statement 1
   6. Abstract 2
   7. Goals and Objectives 2
   8. Relevant mathematics associated with the Project 2
   9. Names of Conferences / Journals where papers can be published 3
   10. Review of Conference/Journal Papers supporting Project idea 3
   11. Plan of Project Execution 6
2. **Technical Keywords** 7
   1. Area of Project 7
   2. Technical Keywords 7
3. **Introduction** 8
   1. Project Idea 8
   2. Motivation of the Project 8
   3. Literature Survey 9
4. **Problem Definition and Scope** 11
   1. Problem Statement 11
      1. Goals and Objectives 11
      2. Statement of Scope 11
   2. Major Constraints 12
   3. Methodologies of Problem solving and efficiency issues 13
   4. Outcome 14
   5. Applications 15
   6. Hardware Resources Required 16
   7. Software Resources Required 16
5. **Project Plan** 17
   1. Project Estimates 18
      1. Reconciled Estimates 18
      2. Project Resources 19
   2. Risk Management w.r.t. NP Hard analysis 20
      1. Risk Identification 21
      2. Risk Analysis 22
      3. Overview of Risk Mitigation, Monitoring, Management 23
   3. Project Schedule 24
      1. Project task set 24
      2. Task Network 25
      3. Timeline Chart 25
   4. Team Organization 26
      1. Team Structure 26
      2. Management Reporting and Communication 26
6. **Software requirement specification** 
   1. Introduction 27
      1. Purpose and Scope of Document 27
      2. Overview of Responsibilities of Developer 27
   2. Usage Scenario 27
      1. User Profiles 28
      2. Use-cases 28
      3. Use Case View 29
   3. Data Model and Description 29
      1. Data Description 29
      2. Data Objects and Relationships 30
   4. Functional Model and Description 30
      1. Data Flow Diagram 31
      2. Activity Diagram: 32
      3. Non-Functional Requirements: 32
      4. State Diagram 33
      5. Design Constraints 34
      6. Software Interface Description 34
7. **Detailed Design Document using Appendix A and B** 35
   1. Introduction 35
   2. Architectural Design 35
   3. Data design (using Appendices A and B) 36
      1. Internal software data structure 36
      2. Global data structure 37
      3. Temporary data structure 37
      4. Database Description 38
   4. Component Design 39
      1. Class Diagram 39
8. **Project Implementation**  40
   1. Introduction 40
   2. Tools and Technologies Used 40
   3. Methodologies/Algorithm Details 41
      1. Algorithms 41
9. **Software Testing**  41
   1. Type of Testing Used 42
   2. Test Cases and Test Results 43
10. **Results**  44
    1. Screenshots 44
    2. Outputs 45
11. **Deployment and Maintenance**  46
    1. Installation and un-installation 46
    2. User help 46
12. **Conclusion and future scope** 47
13. **Bibliography** 48

Appendix References 49

Appendix Project Planner 51

Append Proof Paper Submitted 52

Appendix Plagiarism Report 53

Appendix Information of Project Group Members 54

# List of Figures

|  |  |  |
| --- | --- | --- |
| 6.1 | Task Network | 25 |
| 7.1 | Use Case Diagram | 29 |
| 7.2 | Data objects and Relationships | 30 |
| 7.3 | Level 0 Data Flow Diagram | 31 |
| 7.4 | Level 1 Data Flow Diagram | 32 |
| 7.5 | Activity Diagram | 32 |
| 7.6 | Architecture Diagram | 39 |
| 8.1 | Class Diagram | 40 |
| 11.1 | Registration Form | 45 |
| 11.2 | Login Form | 45 |
| 11.3 | Successful Registration | 46 |
| 11.4 | Successful login | 46 |

VII

# List of Tables

|  |  |  |
| --- | --- | --- |
| 2.1 | Review of Conference/Journal Supporting Project Idea | 5 |
| 2.2 | Plan of Project Execution | 6 |
| 3.1 | Literature Survey | 10 |
| 5.1 | Hardware Requirements | 16 |
| 6.1 | Risk Table | 22 |
| 6.2 | Risk Probability definitions | 22 |
| 6.3 | Risk Impact definitions | 24 |
| 6.4 | Timeline Chart | 25 |
| 7.1 | Use Cases | 28 |

VIII

# **Synopsis**

**2.1 Project Title**

Devanagari Captcha: An Experiment for Interactivity and Security

**2.2 Project Option**

Internal Project

**2.3 Internal Guide**

Mrs. S. T. Somvanshi

**2.4 Technical Keywords (As per ACM Keywords)**

1. CAPTCHA Generation
2. CAPTCHA Verification
3. Challenge-response system
4. Distortion
5. Noise
6. Segmentation
7. Randomization
8. Security

**2.5 Problem Statement**

To generate a Devanagari language-based CAPTCHA.

**2.6 Abstract**

* Devanagari Captcha Generation is a System that generates Captcha in Hindi/Marathi.
* It is a System of text-based CAPTCHA that is based on the Devanagari script .It is written in the

form of Indian languages Hindi and Marathi.

* The objective of a Devnagari CAPTCHA System is to differentiate a human from a Bot. It

can also be used as a language barrier for people who can only understand Hindi and Marathi.

* It is implemented for the better understanding of Indian people. This system will allow people to

understand and use captchas in rural parts.

* Users may use this in any form of login resulting in the swift working of the process.
* This System provides guidelines to improve security control and storage of Captcha methods

against various possible attacks and guidelines to improve their usability.

**2.7 Goals and Objectives**

* The primary purpose of using captchas is to prevent automated bots from accessing a website, and allowing legitimate users to gain access without difficulty.
* Captchas help to ensure that online services remain secure and prevent unauthorized access, such as spamming, scraping, and other malicious activities.

**2.8 Relevant mathematics associated with the Project**

Begin:

Randomly pick a number n = Random();

// captcha module is used

adv(I, Cd , p) = I + {

The immutable adversarial example

}

Discard I;

Randomly select m − 1 different indexes j1,..., jm−1 from[1,..., n]

Choose the representative images [Ij1 ,..., Ijm−1 ] of the corresponding classes;

Output: a random permutation of m possible answers {I, Ij1 ,..., Ijm−1 }.

**2.9 Names of Conferences / Journals where papers can be published**

* IEEE/ACM Conference/Journal 1
* Conferences/workshops in IITs
* Central Universities or SPPU Conferences
* IEEE/ACM Conference/Journal 2
  1. **Review of Conference/Journal Papers supporting Project**

**Idea**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr**  **No** | **Publication**  **& Year** | **Title** | **Description** |
| 1 | IEEE 2020 | Robust CAPTCHAs Towards Malicious OCR | Due to the complexity of OCR model, compared with the common CNN model, standard adversarial training does not show the effectiveness as usual. |
| 2 | IEEE 2019 | A Security Analysis of Captchas With Large Character Sets | The rotation mechanism used in this Captcha scheme negatively impacts the recognition accuracy.    A slight difference between synthetic samples and real data also affects the recognition accuracy. |
| 3 | IEEE 2018 | Vulnerabilities of Existing Design, and Countermeasure. | Each captcha challenge loops continuously within a short video to limit the captcha file size and prevent collection of codeword contours.since each frame is a binary image only black pixes are super imposed to form clearer shape. |
| 4 | IEEE 2018 | Simple and Easy: Transfer Learning-Based Attacks to text CAPTCHA | Using deep leaming techniques, they have tested 20 Roman character-based schemes and 5 Chinese schemes and achieved remarkably good success rates, ranging from 36.3% to 96.9% at an average attack speed 0.02 seconds per CAPTCHA. Most of the tested schemes achieved success rates of 0%. |
| 5 | IEEE 2017 | An End-to-End Attack on Text CAPTCHAs | The variation of character fonts increased the difficulty of the recognition process. This indicates that resisting recognition, rather than resisting segmentation, will be the likely development direction in the future |
| 6 | IEEE 2017 | Selective Learning Confusion Class for  Text-Based-CAPTCHA Recognition | They Test all the class DCNNs with initial testing and then analyse the output by judging whether the prediction result of a testing sample is in the confusion class.  In Bot Detect CAPTCHA, the number of recognized error classes is large and scattered |
| 7 | Springer 2016 | Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha | They proposed the following methodology :-  Generate style-transferred images  Synthesize the background :  Generate the Captcha:  Generate a description.  . |
| 8 | Solar 2016 | No Bot Expects the Deep CAPTCHA!  Introducing Immutable Adversarial Examples,  With Applications to CAPTCHA Generation | The proposed system can operate on the set of labels they have been trained for. Switching to an alternative set of labels is likely to reduce their effectiveness |
| 9 | IEEE 2015 | Captcha as Graphical Passwords—A New Security  Primitive Based on Hard AI Problems. | In entering a CAPTCHA, a user-clicked point is replaced by the grid-cell it lies in. If click errors are in it ,each user-clicked point falls into the same grid-cell as the original CAPTCHA . Therefore the sequence of grid-cells generated from user-clicked points is identical to the one that the authentication server. |
| 10 | ACM  2014 | Chinese Character CAPTCHA Recognition Based on  Convolution Neural Network | In this paper they have studied the efficiency of model training, by conducting experiments to learn how to optimize the training time and to show the effects of different training strategies. Their attack provides a more promising strategy that not only reduces the attack complexity and manual-labeling cost but also preserves comparable accuracy |

**Table 2.1: Review of Conference/Journal Supporting Project Idea**

* 1. **Plan of Project Execution**

|  |  |  |  |
| --- | --- | --- | --- |
| Topic / Module | Module Head | Current Status | Plan of Completion |
| Character Recognition | Mrunal | Completed | August 2021 |
| CAPTCHA Generation | Mrunal | Completed | September 2021 |
| Image Creation | Bhavana | Completed | October 2021 |
| Audio Generation | Bhavana | Completed | October 2021 |
| CAPTCHA Verification | Abhiram | Completed | November 2021 |
| Database Creation | Abhiram | Completed | December 2021 |
| User Interface | Sakshi | Completed | March 2022 |
| Documentation | All Members | Completed | April 2022 |

Figure no. 2.2 Plan of Project Execution

# **Technical Keywords**

**3.1 Area of Project**

CAPTCHA Generation Algorithm, CAPTCHA Security Analysis, Accessibility in Devanagari CAPTCHA, Multi-lingual CAPTCHA Generation, Real-time CAPTCHA Generation and Verification, CAPTCHA Security Enhancement, Human Solvability Evaluation.

**3.2 Technical Keywords**

* CAPTCHA Generation
* CAPTCHA Verification
* Challenge-response system
* Distortion
* Noise
* Segmentation
* Randomization
* Security

# **Introduction**

**4.1 Project Idea**

**Devanagari Captcha: An Experiment for Interactivity and Security**

**4.2 Motivation of the Project**

* **Language-specific protection:** Devanagari captchas can be used to protect websites and online services that primarily cater to users who use the Devanagari script, such as websites in Hindi, Marathi, Nepali, and other languages that use the Devanagari script. By implementing Devanagari captchas, you can ensure that only users who are familiar with the script can access the services, adding an extra layer of security.
* **Cultural relevance:** Devanagari captchas can help promote and preserve the cultural and linguistic identity of communities that use the Devanagari script. By incorporating captchas in their native script, websites can create a more inclusive and familiar user experience for users from those communities
* **Improved user experience:** Captchas are often used to prevent automated bots and malicious activities, but they can sometimes be challenging for users to decipher. By using Devanagari captchas for Devanagari-speaking users, you can provide a more user-friendly experience, as the users will be more accustomed to reading and understanding the script, resulting in quicker and more accurate completion of the captchas.
* **Accessibility:** Devanagari captchas can contribute to making online services more accessible for individuals with visual impairments who use screen readers. By providing captchas in Devanagari, these users can rely on their screen readers to accurately vocalize the content, improving their ability to access and use online platforms.
* **Localized security:** Using Devanagari captchas can be beneficial in regions where English may not be widely understood. By implementing captchas in the local script, you ensure that security measures are aligned with the local user base and minimize potential confusion or barriers caused by language differences

4. Literature survey

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr**  **No** | **Publication**  **& Year** | **Title** | **Description** |
| 1 | IEEE 2020 | Robust CAPTCHAs Towards Malicious OCR | Due to the complexity of OCR model, compared with the common CNN model, standard adversarial training does not show the effectiveness as usual. |
| 2 | IEEE 2019 | A Security Analysis of Captchas With Large Character Sets | The rotation mechanism used in this Captcha scheme negatively impacts the recognition accuracy.  A slight difference between synthetic samples and real data also affects the recognition accuracy. |
| 3 | IEEE 2018 | Vulnerabilities of Existing Design, and Countermeasure. | Each captcha challenge loops continuously within a short video to limit the captcha file size and prevent collection of codeword contours. Since each frame is a binary image only black pixels are super imposed to form clearer shape. |
| 4 | IEEE 2018 | Simple and Easy: Transfer Learning-Based Attacks to text CAPTCHA | Using deep leaming techniques, they have tested 20 Roman character-based schemes and 5 Chinese schemes and achieved remarkably good success rates, ranging from 36.3% to 96.9% at an average attack speed 0.02 seconds per CAPTCHA. Most of the tested schemes achieved success rates of 0%. |
| 5 | IEEE 2017 | An End-to-End Attack on Text CAPTCHAs | The variation of character fonts increased the difficulty of the recognition process. This indicates that resisting recognition, rather than resisting segmentation, will be the likely development direction in the future |
| 6 | IEEE 2017 | Selective Learning Confusion Class for  Text-Based-CAPTCHA Recognition | They Test all the class DCNNs with initial testing and then analyse the output by judging whether the prediction result of a testing sample is in the confusion class.  In Bot Detect CAPTCHA, the number of recognized error classes is large and scattered |
| 7 | Springer 2016 | Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha | They proposed the following methodology :-  Generate style-transferred images  Synthesize the background :  Generate the Captcha:  Generate a description.  . |
| 8 | Solar 2016 | No Bot Expects the Deep CAPTCHA!  Introducing Immutable Adversarial Examples,  With Applications to CAPTCHA Generation | The proposed system can operate on the set of labels they have been trained for. Switching to an alternative set of labels is likely to reduce their effectiveness |
| 9 | IEEE 2015 | Captcha as Graphical Passwords—A New Security  Primitive Based on Hard AI Problems. | In entering a CAPTCHA, a user-clicked point is replaced by the grid-cell it lies in. If click errors are in it ,each user-clicked point falls into the same grid-cell as the original CAPTCHA . Therefore the sequence of grid-cells generated from user-clicked points is identical to the one that the authentication server. |
| 10 | ACM  2014 | Chinese Character CAPTCHA Recognition Based on  Convolution Neural Network | In this paper they have studied the efficiency of model training, by conducting experiments to learn how to optimize the training time and to show the effects of different training strategies. Their attack provides a more promising strategy that not only reduces the attack complexity and manual-labeling cost but also preserves comparable accuracy |

**Table 2.1: Review of Conference/Journal Supporting Project Idea**

# **Problem Definition and scope**

**5.1 Problem Statement**

Devanagari CAPTCHA Generation

## **5.1.1 Goals and Objectives**

**Goal:** Develop an efficient and secure Devanagari Captcha generation system.

**Objectives :**

* Research and analyze existing Devanagari Captcha generation techniques.
* Identify and understand the unique challenges and requirements of generating Devanagari Captchas
* Design and develop a robust algorithm for generating Devanagari Captchas
* Ensure the generated captchas are visually clear, readable, and resistant to automated attacks

## **5.1.2 Statement of Scope**

The scope of this project encompasses the development and evaluation of a Devanagari Captcha generation system with the objective of enhancing security, user experience, and inclusivity for Devanagari-speaking users. The project will focus on the following key aspects:

Devanagari Captcha Generation:

* Design and develop an efficient and secure algorithm for generating Devanagari Captchas
* Research and analyze existing Devanagari Captcha generation techniques.
* Ensure the generated captchas are visually clear, readable, and resistant to automated attacks

User Experience Evaluation:

* Conduct user studies and feedback surveys to assess the usability and effectiveness of the Devanagari Captcha system.
* Measure the accuracy and efficiency of users in completing Devanagari Captchas compared to other captcha types.
* Gather feedback on the clarity and comprehensibility of the generated captchas.

**Accessibility Enhancement:**

* Investigate techniques to make Devanagari Captchas accessible for visually impaired users.
* Test and refine the accessibility features of Devanagari Captchas, ensuring compatibility with screen readers and other assistive technologies.
* Ensure the generated captchas are perceivable and understandable through alternative means for visually impaired users.

**Adoption and Integration:**

* Raise awareness about the benefits of using Devanagari Captchas for Devanagari-speaking communities.
* Collaborate with web developers and platform owners to integrate the Devanagari Captcha system into their websites or services.
* Provide documentation and resources for easy implementation and customization of the Devanagari Captcha system

**The project does not include:**

* Extensive modifications or adaptations of the Devanagari script itself.
* Integration with specific platforms or websites beyond collaborative efforts with interested parties.

**5.2 Major Constraints**

**Language Complexity:** The Devanagari script has a complex character set with various ligatures, diacritic marks, and character combinations. Developing a Devanagari Captcha system requires addressing these complexities to ensure accurate and visually clear captchas.

**Font and Rendering Compatibility:** Devanagari fonts and their rendering can vary across different devices, browsers, and operating systems. Ensuring consistent and accurate rendering of the captchas across various platforms and configurations may pose challenges.

**User Acceptance and Usability:** The project should take into account user preferences, feedback, and cultural considerations related to the usage of Devanagari captchas. Balancing security requirements with user-friendly and intuitive designs is crucial for user acceptance and engagement.

**Collaboration and Integration:** Collaborating with website owners, platform developers, or CMS providers to integrate the Devanagari Captcha system may face challenges due to diverse technologies, compatibility issues, and varying levels of adoption.

**Resource Limitations:** The project's scope and execution may be influenced by resource constraints, including time, funding, development expertise, and availability of language-specific datasets for training and testing the algorithm.

**Localization and Internationalization:** Adapting the Devanagari Captcha system for different languages and scripts within the Devanagari family (such as Hindi, Marathi, and Nepali) may require additional considerations and efforts.

**5.3 Methodologies of Problem-solving and efficiency issues**

**Methodologies of Problem-Solving**:

**Define the Problem:** Clearly articulate and understand the problem statement related to Devanagari Captcha generation, user experience, accessibility, or integration. Identify the specific challenges and constraints involved.

**Research and Analysis:** Conduct thorough research and analysis of existing techniques, algorithms, and best practices related to Devanagari Captcha generation, usability, accessibility, and integration. Study academic papers, industry standards, and relevant resources to gain insights into effective solutions.

**Brainstorming and Ideation:** Engage in brainstorming sessions to generate innovative ideas and potential solutions to the identified problem. Encourage collaboration and diverse perspectives to explore different possibilities.

**Design and Prototyping**: Develop a systematic and well-thought-out design for the Devanagari Captcha system. Create prototypes or mock-ups to visualize and refine the proposed solution. Iterate on the design based on feedback and evaluation.

**Implementation and Testing:** Implement the designed solution, considering efficiency, security, and usability. Conduct rigorous testing to identify and resolve any issues or bugs. Test the system under various scenarios and validate its effectiveness.

**Evaluation and Feedback:** Evaluate the implemented solution through user studies, feedback surveys, and performance metrics. Gather feedback from users, experts, and stakeholders to assess the efficiency, effectiveness, and user satisfaction of the Devanagari Captcha system.

**Iterative Improvement**: Based on the evaluation results and feedback received, make iterative improvements to the system. Address any identified efficiency issues, security vulnerabilities, usability challenges, or accessibility concerns.

**Efficiency Issues:**

**Algorithm Optimization:** Ensure that the Devanagari Captcha generation algorithm is optimized for efficiency. Consider techniques like caching, pre-computation, or parallelization to reduce computational overhead and improve response times.

**Resource Management:** Efficiently manage system resources such as memory, processing power, and storage to minimize resource consumption. Avoid unnecessary redundancy or inefficient data structures that can impact performance.

**Scalability and Performance:** Design the system to handle increased user load and scale horizontally or vertically as needed. Implement performance monitoring and profiling to identify bottlenecks and optimize critical components.

**Minimize Network Latency:** Optimize network communication and reduce latency by employing techniques such as data compression, content delivery networks (CDNs), or efficient data transfer protocols.

**Caching and Response Caching:** Utilize caching mechanisms to store and reuse frequently accessed data, such as generated captchas or validation results. This can significantly reduce the computational overhead and improve response times.

**Code Efficiency**: Write clean and efficient code by following best practices, employing appropriate data structures and algorithms, and avoiding unnecessary computations or redundant operations.

**Continuous Monitoring and Optimization:** Regularly monitor system performance, analyze metrics, and identify areas for optimization.

**5.4 Outcome**

**Developed Devanagari Captcha Generation System:** The project will result in the development of an efficient and secure algorithm for generating Devanagari Captchas. The system will be capable of producing visually clear and readable captchas that are resistant to automated attacks.

**Improved User Experience:** The Devanagari Captcha system will enhance the user experience for Devanagari-speaking individuals. Users will find the captchas more familiar and easier to read, leading to quicker and more accurate completion of the captchas.

**Enhanced Accessibility:** The project will address accessibility concerns by ensuring compatibility with screen readers and other assistive technologies. Visually impaired users will be able to perceive and understand the Devanagari Captchas through alternative means, improving their accessibility to online platforms.

**Increased Security:** The generated Devanagari Captchas will provide a robust layer of security against automated attacks, such as OCR or machine learning algorithms. This will help protect websites and online services that implement Devanagari Captchas from malicious activities and unauthorized access.

**Adoption and Integration:** The project aims to promote the adoption of Devanagari Captchas in relevant online platforms and websites. Collaborative efforts with developers and platform owners will result in the integration of the Devanagari Captcha system, expanding its usage and impact.

**Cultural Relevance and Preservation:** By incorporating captchas in the Devanagari script, the project will contribute to the cultural relevance and preservation of Devanagari-speaking communities. It will create a more inclusive and familiar online environment for users from those communities.

**Documentation and Resources:** The project will provide documentation and resources for easy implementation and customization of the Devanagari Captcha system. This will support developers and platform owners in implementing the captchas effectively and efficiently

* 1. **Applications**
* **Websites and Online Portals:** Devanagari Captchas can be implemented in websites and online portals that provide content, services, or transactions in Devanagari-based languages such as Hindi, Marathi, Nepali, etc. This can include e-commerce platforms, news websites, educational portals, government websites, and more.
* **User Registration and Login**: Devanagari Captchas can be used as an additional security measure during user registration and login processes. By verifying that users can accurately decipher the Devanagari captchas, it helps prevent automated bots or unauthorized access to user accounts.
* **Online Forms and Surveys:** Devanagari Captchas can be integrated into online forms and surveys to ensure that responses are generated by genuine human users. This helps prevent spam submissions and ensures data integrity.
* **Account Recovery and Verification:** Devanagari Captchas can be employed in account recovery or verification processes, such as password resets or email verifications. By including Devanagari captchas, adds an extra layer of security to validate the identity of users.
* **Online Voting and Polls:** Devanagari Captchas can be utilized in online voting systems and polls conducted in Devanagari-based languages. This helps ensure that the participants are genuine human users, preventing automated voting or manipulation.
* **Mobile Applications:** Devanagari Captchas can be incorporated into mobile applications that are designed for Devanagari-speaking users. This can include various types of apps such as language learning apps, news apps, social media apps, and more.
* **Content Accessibility:** Implementing Devanagari Captchas can contribute to making online content more accessible to individuals who are fluent in Devanagari-based languages. It ensures that access to content, features, or services is limited to users who can read and understand the script.
  1. **Hardware Resources Required**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameter | Minimum Requirement | Justification |
| 1 | CPU Speed | 2 GHz | Remark Required |
| 2 | RAM | 3 GB | Remark Required |

Table 5.1: Hardware Requirements

* 1. **Software Resources Required**

**Platform :**

* Programming Language : Python
* Integrated Development Environment (IDE): Visual Studio
* Image Processing Libraries:
* Web Frameworks: Flask , ReactJs
* Version Control: Git
* Database Management System: MySql
* Documentation and Collaboration Tools

# **Project Plan**

**6.1 Project Estimates**

**Requirements Gathering and Analysis**: Define the specific requirements of the Devanagari Captcha system. Determine the specific **Devanagari-based language**(s) you want to support, such as Hindi, Marathi, Nepali, etc. Identify the characters and symbols within the Devanagari script that will be used in the captcha generation., including security, usability, accessibility, and integration aspects. Conduct a detailed analysis of existing captcha systems, algorithms, and best practices.

**System Design:** Design the architecture and components of the Devanagari Captcha system, considering scalability, efficiency, and security. Define the captcha generation algorithm and image rendering techniques. Design the user interface and user experience (UI/UX) aspects, ensuring readability and accessibility for Devanagari-speaking users.

**Implementation:** Develop the Devanagari Captcha system according to the defined requirements and design specifications. Implement the captcha generation algorithm, image rendering techniques, and security measures. Develop any necessary backend systems or APIs for captcha validation and integration. We designed our frontend with ReactJs, the backend with flask and have used PHP as our database.

**Testing:** Conduct comprehensive testing of the Devanagari Captcha system to ensure its functionality, usability, and security. Perform unit testing, integration testing, and system testing to identify and resolve any issues or bugs. Validate the system's performance under various scenarios and user load conditions.

**Documentation:** Create documentation for the Devanagari Captcha system, including technical specifications, user guides, and implementation instructions. Document the algorithms, design choices, and system configurations for future reference and maintenance.

**Deployment and Integration**: Collaborate with website owners, platform developers, or CMS providers to integrate the Devanagari Captcha system into relevant platforms or websites. Coordinate the deployment and integration process, ensuring compatibility and smooth adoption.

**User Feedback and Evaluation:** Gather user feedback on the usability, readability, and effectiveness of the Devanagari Captcha system. Evaluate the system's performance and security by analyzing user feedback, system logs, and performance metrics. Incorporate user feedback and make necessary improvements to enhance the system.

## **Maintenance and Support:** Provide ongoing maintenance and support for the Devanagari Captcha system, addressing any issues or security vulnerabilities that may arise. Stay updated with emerging technologies, security threats, and best practices related to captchas and implement necessary updates and enhancements

## **6.1.1 Reconciled Estimates**

**6.1.1.1 Cost Estimate**

**Development Costs:** Salaries and wages for the development team, including developers, designers, and researchers. Costs associated with hiring or outsourcing development resources if needed. Licensing fees for any software or libraries used in the development process. Costs for development tools and software.

**Testing and Quality Assurance Costs:** Costs associated with testing the Devanagari Captcha system, including manual testing and automated testing tools. Expenses for bug fixing, code review, and quality assurance processes.

**Documentation and Support Costs**: Costs for creating documentation, user guides, and help resources. Support and maintenance costs, including ongoing updates, bug fixes, and user support.

**Integration and Deployment Costs:** Expenses related to integrating the Devanagari Captcha system into existing platforms or websites. Costs associated with deployment, configuration, and setup.

**Miscellaneous Costs:** Costs for project management, including coordination, communication, and stakeholder meetings. Marketing and promotion costs, if applicable, to raise awareness of the Devanagari Captcha system.

**6.1.1.2 Time Estimates**

* Requirements Gathering and Analysis: 1-2 weeks
* System Design: 1-2weeks
* Implementation: 4-6 weeks
* Testing: 2-4 weeks
* Documentation: 1-2 weeks Deployment and Integration: 1-2 weeks
* User Feedback and Evaluation: 2-4 weeks
* Maintenance: Ongoing

## **6.1.2 Project Resources**

**Human Resources:** Development Team: This includes software developers, designers, and researchers who will be responsible for developing the captcha generation algorithm, designing the user interface, and implementing the system.

* **Project Manager:** A project manager will oversee the project, coordinate activities, and ensure that the project progresses according to the plan.
* **Quality Assurance/Testers:** These individuals will be responsible for testing the Devanagari Captcha system, identifying and reporting bugs, and ensuring the system's quality.
* **Documentation Specialist:** A documentation specialist will create user guides, technical documentation, and other project-related documentation.

**Hardware Resources:** Development Machines: Each team member will require a computer with sufficient processing power, memory, and storage to develop the captcha system.

* **Server Infrastructure:** If you plan to deploy the captcha system on your own servers, you will need hardware resources such as servers, storage devices, and networking equipment.
* **Software and Tools:** Integrated Development Environment (IDE): Developers will require an IDE to write, debug, and test the code. Popular IDEs include Visual Studio, Eclipse, or JetBrains IntelliJ IDEA.
* **Design Software:** Designers will use tools such as Adobe Photoshop, Sketch, or Figma to create visual elements and design the user interface.
* **Version Control System:** Utilize a version control system like Git to manage code versions and facilitate collaboration among team members.
* **Testing Tools:** Choose appropriate testing tools, such as unit testing frameworks, automated testing tools, or load testing tools, to ensure the quality of the captcha system.

**Data and Datasets:**

* **Devanagari Character Dataset**: You may need a dataset of Devanagari characters for training and testing the captcha generation algorithm. This dataset can be obtained from public sources or created by collecting and annotating Devanagari characters.

**Documentation Tools:** Utilize documentation tools like Microsoft Word, Google Docs, or Confluence to create and manage project-related documents.

**Financial Resources:**

* **Budget:** Allocate a budget to cover various project expenses, including human resources, hardware resources, software licenses, and any other costs associated with the project.

**6.2 Risk Management w.r.t. NP-Hard analysis**

**Technical Risks:**

## **Risk:** Inaccurate or inefficient captcha generation algorithm.

## **Approach:** Conduct thorough research and analysis of existing captcha generation algorithms. Implement and test multiple algorithms, evaluating their accuracy, efficiency, and suitability for Devanagari characters. Iterate and refine the algorithm based on performance evaluation.

## **Security Risks:**

## **Risk:** Vulnerabilities in the captcha system exploited by automated bots.

## **Approach:** Implement security measures such as character distortion, noise addition, or background variations to prevent automated recognition. Regularly monitor and update the captcha system to address emerging security threats. Conduct security audits and penetration testing to identify and rectify any vulnerabilities.

## **Scalability Risks:**

## **Risk:** Inadequate scalability to handle increasing user demand.

## **Approach:** Design the system with scalability in mind from the beginning. Utilize scalable server architectures, distributed computing, and caching mechanisms. Monitor system performance and scalability regularly, and make necessary optimizations and infrastructure adjustments as the user load increases.

## **Resource Risks:**

## **Risk:** Insufficient availability of development resources.

## **Approach:** Plan resource allocation carefully, considering the project timeline and required skill sets. Identify potential bottlenecks in resource availability and plan for backup resources or external support if necessary. Maintain clear communication and coordination among team members to optimize resource utilization.

## **Integration Risks:**

## **Risk:** Challenges in integrating the captcha system into existing platforms or websites.

## **Approach:** Conduct thorough compatibility testing with target platforms and ensure proper documentation and guidelines for integration. Collaborate closely with platform developers or CMS providers to address any integration issues. Provide clear documentation and support to facilitate smooth integration.

## **Time and Schedule Risks:**

## **Risk:** Delays in project milestones or missed deadlines.

## **Approach:** Create a realistic project schedule with buffer time for unforeseen delays. Break down the project into manageable tasks with clear timelines and dependencies. Regularly monitor project progress and address any issues or roadblocks promptly.

## **Budget Risks:**

## **Risk:** Exceeding the allocated budget due to unforeseen expenses.

## **Approach:** Develop a detailed budget plan, accounting for all project expenses. Monitor and track expenses throughout the project, making adjustments as needed. Conduct regular financial reviews and communicate with stakeholders to ensure transparency and address any budgetary risks proactively

## **6.2.1 Risk Identification**

Risk identification is an essential step in project risk management. Here are some potential risks to consider when developing a Devanagari Captcha system:

**Technical Risks:** Inaccurate or inefficient captcha generation algorithm. Inadequate performance and scalability of the system. Compatibility issues with different browsers or devices. Inability to handle a large number of simultaneous users.

**Security Risks:** Vulnerabilities in the captcha system exploited by automated bots. Unauthorized access or breaches compromising user data. Inadequate encryption or data protection measures. Resource Risks: Insufficient availability of skilled development resources. Lack of necessary hardware or software infrastructure. Delays or disruptions due to external dependencies (e.g., third-party APIs, libraries).

**Integration Risks:** Challenges in integrating the captcha system into existing platforms or websites. Incompatibility with specific content management systems (CMS) or frameworks. Difficulties in aligning with the user experience and design of the target platform.

**Schedule Risks:** Unrealistic project timeline or milestones. Dependencies on external factors or deliverables. Delays in requirements gathering or changes in project scope.

**Budget Risks:** Insufficient budget allocation for development, resources, and infrastructure. Unforeseen expenses arising during the project. Cost overruns due to changes in requirements or scope.

**User Acceptance Risks:** Poor usability or accessibility of the captcha system. Lack of user adoption or resistance to using captchas. Negative feedback or dissatisfaction from users.

**Legal and Compliance Risks:** Non-compliance with privacy regulations or data protection laws. Violation of intellectual property rights or copyright infringements. Failure to meet accessibility standards or regulations

## **6.2.2 Risk Analysis**

The risks for the Project can be analyzed within the constraints of time and quality

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Risk Description | Probability |  | Impact |  |
| Schedule | Quality | Overall |
| 1 | Security Risk | Low | Low | High | High |
| 2 | Integration Risk | Low | Low | High | High |

**Table 6.1: Risk Table**

|  |  |  |
| --- | --- | --- |
| Probability | Value | Description |
| High | Probability of occurrence is | *>* 75% |
| Medium | Probability of occurrence is | 26−75% |
| Low | Probability of occurrence is | *<* 25% |

**Table 6.2: Risk Probability definitions**

## **6.2.3 Overview of Risk Mitigation, Monitoring, and Management**

**Risk Mitigation:** Identify high-priority risks that pose significant threats to the project and its objectives. Develop and implement risk mitigation strategies to reduce the likelihood and impact of these risks.

* **Example mitigation strategies:** Conduct thorough research and analysis of existing captcha generation algorithms to minimize the risk of inaccurate or inefficient algorithms.
* Implement security measures such as character distortion and noise addition to prevent automated bot recognition.
* Use scalable server architectures and caching mechanisms to handle increasing user demand.
* Plan for backup development resources or external support to mitigate resource availability risks.

**Risk Monitoring:** Continuously monitor identified risks throughout the project lifecycle. Regularly assess the effectiveness of risk mitigation strategies and their impact on the project. Keep track of any changes in the project environment that may affect the identified risks.

* **Example monitoring activities:** Conduct regular code reviews and security audits to identify potential vulnerabilities in the captcha generation system.
* Monitor system performance and scalability as user demand increases, adjusting resources and infrastructure as needed.
* Stay updated on emerging security threats and apply necessary updates and patches to the captcha system.

**Risk Management:** Establish a risk management plan that outlines the overall approach to identifying, analyzing, and responding to risks. Assign responsibilities to team members for managing specific risks and their associated mitigation strategies. Regularly communicate and document the status of risks to stakeholders, ensuring transparency and accountability.

* **Example risk management activities:** Conduct regular risk assessments and update the risk register with new risks or changes to existing risks.
* Review and update the risk mitigation strategies as needed based on ongoing monitoring and evaluation. Continuously engage with stakeholders and address any concerns or issues related to the risks identified in the project

**6.3 Project Schedule**

## **6.3.1 Project task set**

Major Tasks in the Project stages are:

|  |  |  |
| --- | --- | --- |
| Impact | Value | Description |
| Very high | *>* 10% | Text Generation |
| High | 5−10% | Image Generation |
| Medium | *<* 5% | Character Segmentation and Distortion and Noise Application |

**Table 6.3: Risk Impact definitions**

* Task 1: Text Generation
* Task 2: Image Generation
* Task 3: Character Segmentation
* Task 4: Distortion and Noise Application
* Task 5: Accessibility Considerations
* Task 6: Randomization and Variation
* Task 7: Security Analysis
* Task 8: Usability and User Experience
* Task 9: Integration
* Task 10: Testing and Evaluation

**6.3.2 Task network**

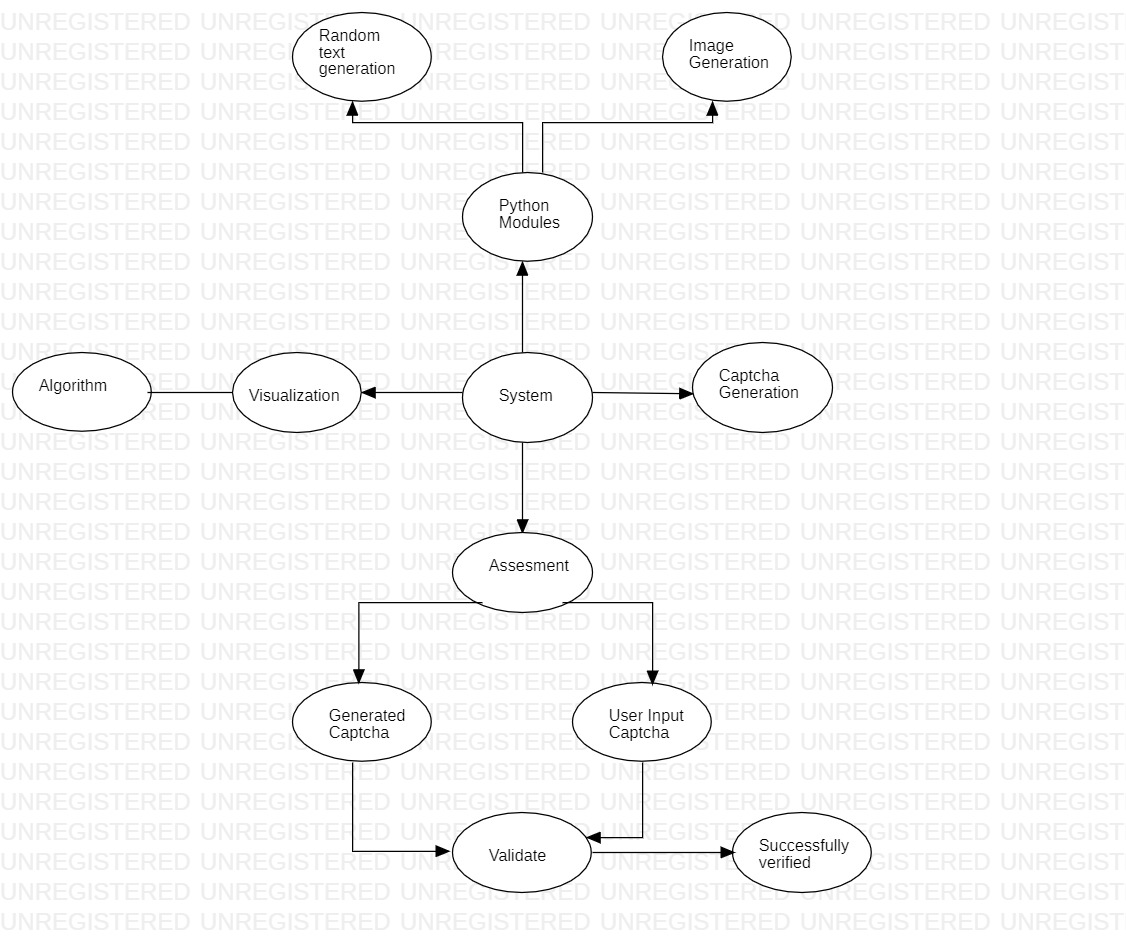


Fig no. 6.1 Task Network

## **6.3.3 Timeline Chart**

|  |  |  |
| --- | --- | --- |
| Topic / Module | Current Status | Plan of Completion |
| Character Recognition | Completed | September 2021 |
| CAPTCHA Generation | Completed | October 2021 |
| Image Creation | Completed | December 2021 |
| Audio Generation | Completed | February 2022 |
| CAPTCHA Verification | Completed | March 2022 |
| Database Creation | Completed | April 2022 |

Table no. 6.4 Timeline Chart

**6.4 Team Organization**

**Project Guide** : Prof. Mrs. S. T. Somvanshi

**Image and audio creation** : Bhavana Thakare

**Character and captcha generation** : Mrunal Sadawarte

**Captcha Verification** : Abhiram Shelkar

**User Interface** : Sakshi Ghonge

**6.4.1 Team structure**

* **Bhavana Thakare** : Responsible for Image recognition and Audio Generation.
* **Mrunal Sadawarte :** Responsible for Character Recognition and CAPTCHA generation.
* **Abhiram Shelkar :** Responsible for CAPTCHA verification and Database design.
* **Sakshi Ghonge :** Responsible for User-Interface and Documentation.

## **6.4.2 Management reporting and communication**

Google Drive is used for reporting and keeping all work in sync with all members of the group

# **Software requirement specification**

**7.1 Introduction**

**7.1.1 Purpose and Scope of Document**

The purpose of Devanagari CAPTCHA is to differentiate between human users and automated bots or computer programs. It aims to ensure that certain actions or access to resources are performed only by humans, preventing malicious activities such as spamming, fraud, or unauthorized access.

**Scope :**

* **User Verification:** Devanagari CAPTCHAs are used to verify that a user interacting with a system or website is human and not a bot.
* **Enhancing Security:** Devanagari CAPTCHAs contribute to the overall security of systems and applications by reducing the risk of unauthorized access or exploitation.
* **Accessibility Considerations:** Alternative text descriptions, audio-based challenges, or other accessible features can be integrated to accommodate users with visual impairments or other disabilities.
* **User Experience:** CAPTCHAs should be designed to be user-friendly, with clear instructions, legible characters, and an appropriate level of difficulty to minimize user frustration.

## 7.1.2 **Overview of responsibilities of Developer**

The developers have extensively worked on Captcha Generation in Devanagari Language and audio generation. These modules are then merged and bound together with the help of a user-friendly interface.

**7.2 Usage Scenario**

**Online Account Registration:** During the registration process for online platforms, Devanagari CAPTCHAs can be utilized to verify that the user signing up is a human, preventing automated bots from creating numerous fake accounts.

**Comment or Forum Submission**: Devanagari CAPTCHAs can be employed when users submit comments or posts on websites or forums. By requiring users to solve a CAPTCHA challenge, it helps prevent spam or malicious content generated by automated bots.

**Password Recovery:** When users request a password reset or recovery, Devanagari CAPTCHAs can be integrated to ensure that the request is initiated by a genuine human user and not an automated script attempting to gain unauthorized access.

**Online Surveys or Voting:** Devanagari CAPTCHAs can be implemented in online surveys or voting systems to ensure that only real users participate and provide their input, preventing automated scripts from manipulating results or skewing data.

**7.2.1 User profiles**

The profiles of all user categories are described here. (Actors and their Description)

**User:** Users input the characters or symbols displayed in the Devanagari CAPTCHA image to prove that they are human and not a computer program attempting to automate a task.

**Developer:** The developer can add & modify functionalities based on user feedback from time to time to make the software more precise, accurate, and helpful to users.

## **7.2.2 Use-cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr No. | Use Case | Description | Actors | Assumptions |
| 1 | User Registration with Devanagari CAPTCHA | The use case involves a user registering for an account on a website that utilizes a Devanagari CAPTCHA for security purposes. | Website User: The individual who intends to create an account on the website. | The user has access to the website's registration page.  The user has basic knowledge of Devanagari characters and can interpret them. |
| 2 | Use Case: Login Authentication with Devanagari CAPTCHA | The use case involves a user logging into an account on a website that employs a Devanagari CAPTCHA as part of the authentication process. | Website User: The individual attempting to log into their account on the website. | The user has an existing account on the website.  The user knows their login credentials (username/email and password). |

Table 7.1: Use Cases

## 7.2.3 Use Case View

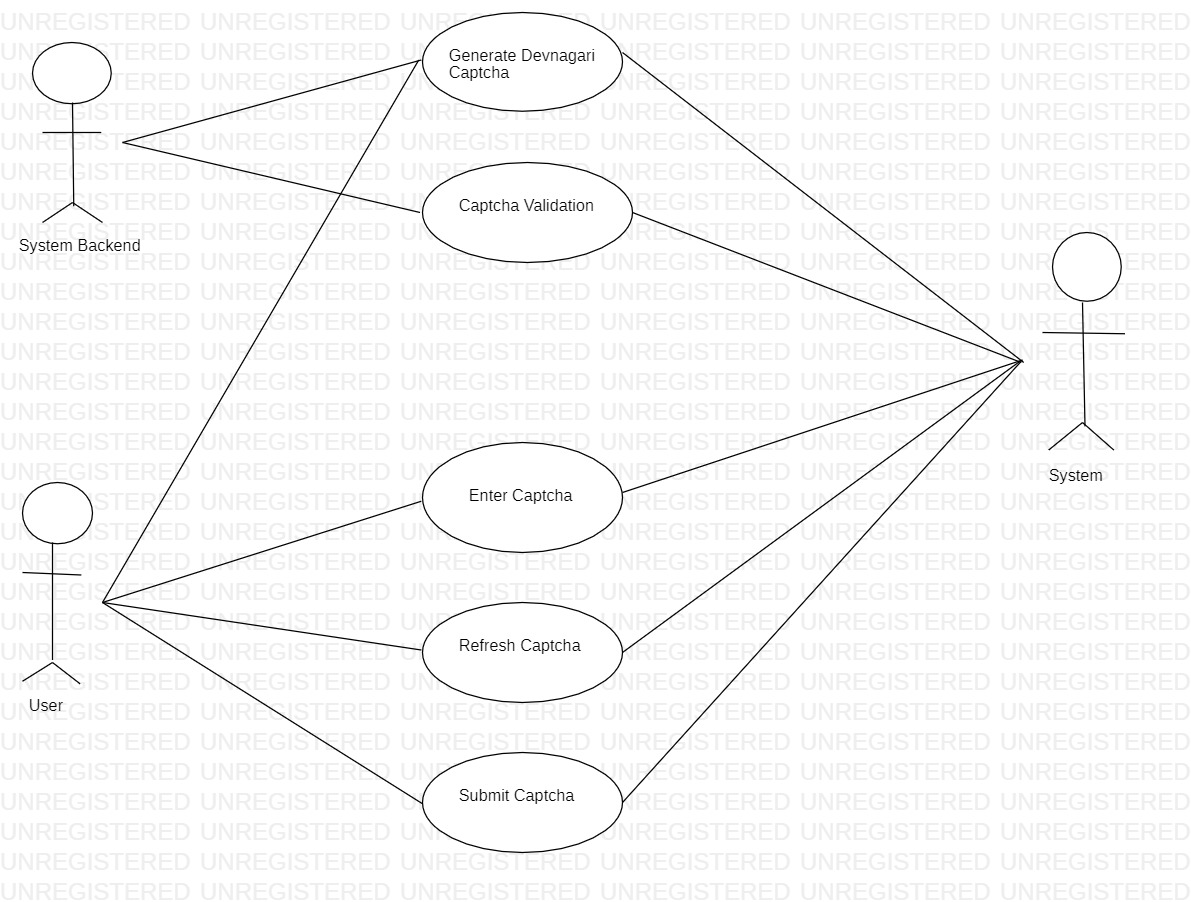


Figure 7.1: Use case diagram

**7.3 Data Model and Description**

## **7.3.1 Data Description**

## **Devanagari Characters:**

## The dataset should include a collection of Devanagari characters. Devanagari is the script used for writing several languages, including Hindi, Nepali, and Sanskrit. The characters range from basic consonants and vowels to complex conjuncts.

## Ensure that the dataset covers a diverse range of Devanagari characters, including all the major consonants, vowels, and conjuncts.

## **7.3.2 Data objects and Relationships**

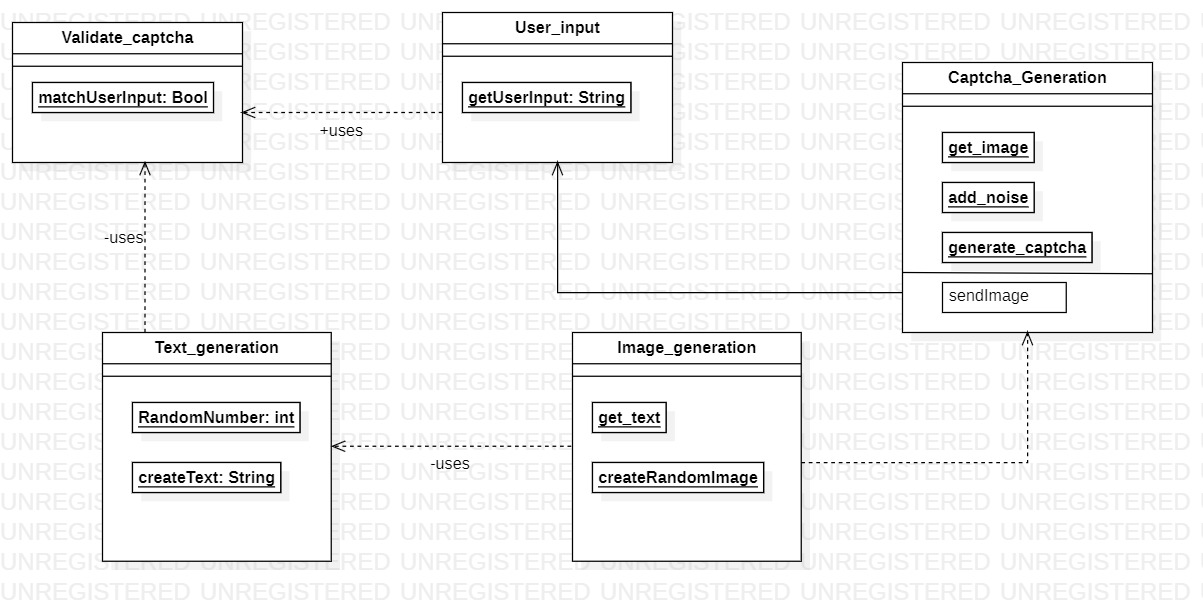


Figure no. 7.2 Data objects and Relationships

**7.4 Functional Model and Description**

The class diagram depicts the relationships between all functions, modules, and data structures, as well as the ties between them, such as extends and aggregation. The Captcha\_Generation generates a random string of a particular length and generates a image for the generated string. The functional model outlines the steps involved in generating a Devanagari CAPTCHA. The model takes inputs such as the desired length of the CAPTCHA, a dataset of Devanagari characters with corresponding images, and an image-character mapping. It also allows for optional parameters to customize the CAPTCHA appearance.

## **7.4.1 Data Flow Diagram**

**7.4.1.1 Level 0 Data Flow Diagram**

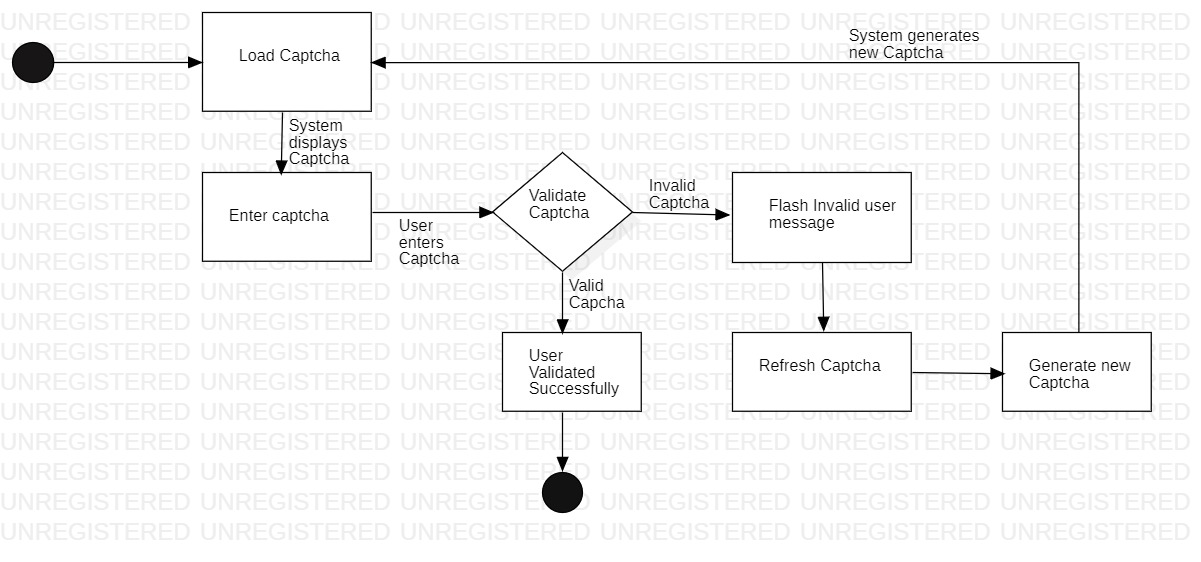
Input : Devnagari Characters

Generation of Captcha and verification

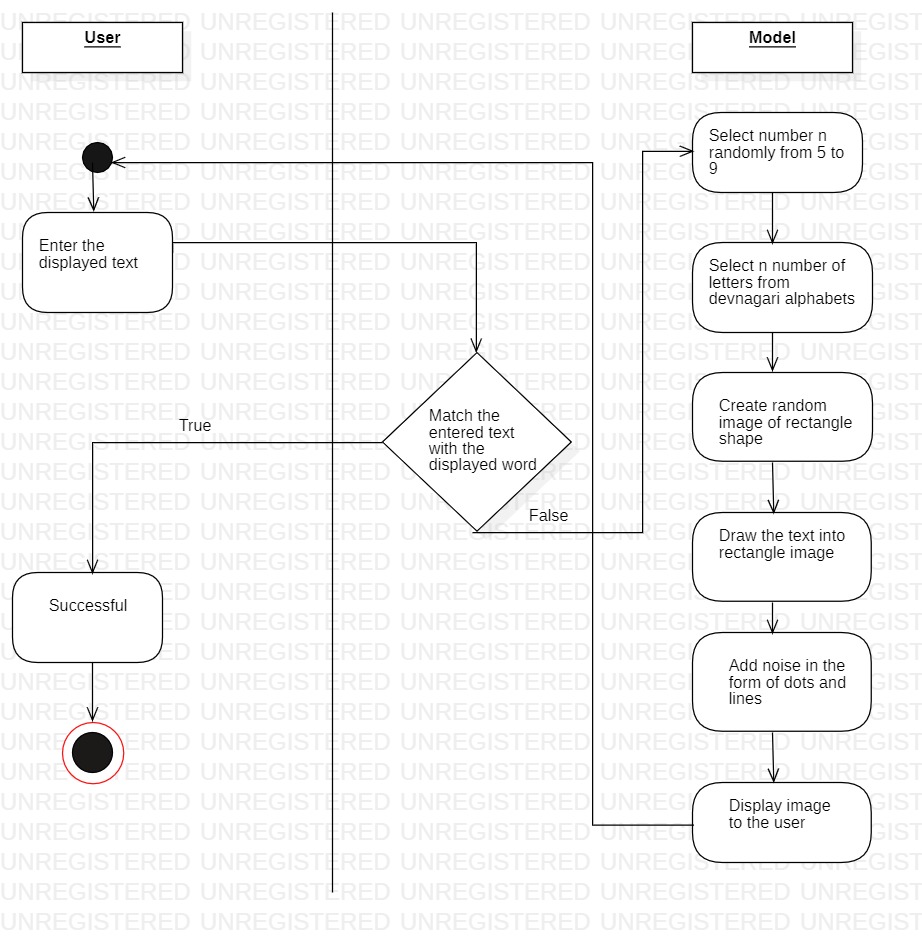
Output: Devanagari Captcha

MySql Database

### **7.4.1.2 Level 1 Data Flow Diagram**



**7.4.2 Activity Diagram:**

****

7.4.3 **Non Functional Requirements:**

**Performance:**

* Efficiency: The CAPTCHA generation process should be optimized to generate CAPTCHAs quickly, minimizing any delays in user interactions.

**Security:**

* Robustness: The generated CAPTCHAs should be robust against automated attacks, making it difficult for bots or computer algorithms to solve them.
* Randomness: The selection and arrangement of characters should be random and unpredictable to prevent pattern recognition or algorithmic exploitation.
* Resistance to OCR: The CAPTCHAs should be designed to be resistant to Optical Character Recognition (OCR) techniques used by automated bots.

**Usability:**

* Readability: The generated CAPTCHAs should be visually clear and legible, ensuring that human users can easily interpret and enter the characters.
* Accessibility: The CAPTCHAs should be accessible to users with visual impairments or other disabilities by providing alternative methods, such as audio-based CAPTCHAs or alternative text prompts.
* User Experience: The CAPTCHA generation process should be user-friendly, providing clear instructions and feedback to users during the interaction.

**Scalability:**

* Performance under Load: The CAPTCHA generation system should be able to handle a high volume of requests efficiently, ensuring smooth functioning during peak loads.
* Horizontal Scalability: The system should be designed to scale horizontally by adding additional resources or servers to handle increased demand.

**Maintainability:**

* Modularity: The CAPTCHA generation system should be designed in a modular and extensible manner, allowing for easy maintenance and future enhancements.
* Code Maintainability: The codebase for CAPTCHA generation should follow best practices, maintain readability, and be well-documented to facilitate ongoing maintenance and updates.

**Compatibility:**

* Cross-platform Compatibility: The CAPTCHA generation system should be compatible with different platforms and devices, ensuring consistent performance and functionality.

**7.4.4 State Diagram:**

**State Transition Diagram**

A state diagram is a graphical depiction of a state machine and one of the 14 different types of UML diagrams for software and systems. A behavioral model made up of states, state transitions, and actions is depicted in state diagrams. The allowable states and transitions, as well as the events that affect these transitions, are depicted in state diagrams.

**7.4.5 Design Constraints**

## **Character Set Limitations:**

## Limited Character Set: The CAPTCHA generation system must work within the constraints of the available Devanagari character set. It should be designed to handle a specific set of characters and not rely on characters outside of this set.

## Character Consistency: The generated CAPTCHAs should maintain consistency in terms of character style, font, and size to ensure a coherent visual representation.

## **Image Generation Constraints:**

## Image Size and Resolution: The CAPTCHA generation system should adhere to predefined size and resolution constraints to ensure consistency and compatibility across different devices and platforms.

## **Security Constraints:**

## Security Measures: The CAPTCHA generation system should implement appropriate security measures to protect against attacks, such as rate limiting, IP blocking, or detection of suspicious activity.

## Avoiding Patterns: The system should be designed to prevent the formation of predictable patterns or repetitions within the generated CAPTCHAs, as this could potentially be exploited by automated algorithms.

## **Performance Constraints:**

## Response Time: The CAPTCHA generation system should generate CAPTCHAs within an acceptable response time to provide a seamless user experience.

## Scalability: The system should be designed to handle a large volume of CAPTCHA requests simultaneously and efficiently, without compromising performance.

## **7.4.6 Software Interface Description**

**User Interface:**

The user interface allows users to interact with the CAPTCHA generation system. It can include web-based forms, APIs, or command-line interfaces.

The user interface provides input fields or parameters for specifying the desired CAPTCHA length, font style, image size, or any other customization options.

It may also display the generated CAPTCHA image to users for visual verification or provide alternative options for accessibility, such as audio-based CAPTCHAs.

# **Detailed Design Document using Appendix**

# **A and B**

**8.1 Introduction**

The purpose of this design document is to outline the implementation details and specifications for generating Devanagari Captchas. Captchas (Completely Automated Public Turing tests to tell Computers and Humans Apart) are widely used in web applications to verify that a user is human and not a malicious bot. Devanagari, the script used for writing languages such as Hindi, Nepali, and Marathi, poses a unique challenge for captcha generation due to its complex and varied character forms.

This document aims to provide a comprehensive understanding of the design considerations, architecture, and algorithms involved in the generation of Devanagari Captchas. It will cover the key objectives, the target audience, and the high-level overview of the proposed solution

**8.2 Architectural Design**

**Random Character Generation:** This component generates random Devanagari characters based on predefined patterns and rules. It ensures the readability and diversity of the characters generated for Captchas.

**Image Rendering:** The generated characters are rendered onto images using appropriate fonts, colors, and backgrounds. This component focuses on producing visually distinct and distortion-resistant Captcha images that are easy for humans to read but challenging for automated algorithms to decipher.

**Noise and Distortion**: To enhance the security and complexity of the Captchas, this component applies various noise and distortion techniques to the rendered images. These techniques may include adding random lines, dots, warping effects, or other visual perturbations that make it difficult for automated algorithms to accurately recognize the characters.

**Validation and Verification:** The validation and verification component ensures the correctness of user-entered Captchas. It compares the user's input with the original Captcha and determines its accuracy. This component may include algorithms for fuzzy matching or other techniques to account for minor variations in user input.

**8.3 Data design (using Appendices A and B)**

**Appendix A: Devanagari Character Set**

Appendix A provides a comprehensive list of Devanagari characters, including vowels, consonants, digits, and additional symbols. Each character has a unique Unicode value associated with it.

For the data design, we can create a data structure to store the Devanagari character set, which could consist of the following attributes:

* **Character:** The Devanagari character itself.
* **Unicode:** The Unicode value representing the character.

This data structure can be implemented as a simple table or a collection, depending on the specific requirements of the programming language or database being used. It will serve as a reference for generating random characters during the captcha generation process.

**Appendix B: Devanagari Character Attributes**

Appendix B provides additional information about each Devanagari character, such as its associated vowels, consonants, and symbols. This information can be useful for ensuring the correct formation of valid Captchas.

To incorporate this information into the data design, we can create a separate data structure to represent the attributes of Devanagari characters. This structure could include the following attributes:

* **Character:** The Devanagari character itself.
* **Vowels**: A list of associated vowels (if applicable).
* **Consonants:** A list of associated consonants (if applicable).
* **Symbols:** A list of associated symbols (if applicable).

**8.3.1 Internal software data structure**

**Captcha Configuration:** This data structure stores the configuration settings for generating Captchas. It includes attributes such as the length of the Captcha, the allowed character set, the complexity level, and any additional customization options.

**Captcha Image:** This data structure represents the generated Captcha image. It typically includes attributes such as the image data, dimensions, file format, and any metadata associated with the image.

**Captcha Solution:** This data structure stores the correct solution to a generated Captcha. It is used for validation and verification purposes when comparing user input with the original Captcha.

**User Response:** This data structure holds the user's response to a presented Captcha. It includes the user's input, timestamps, and any metadata associated with the user's interaction.

**Session Data:** If the Captcha generation system operates within a session-based environment, a data structure can be used to store session-specific information. This can include session IDs, session start and end times, and any other relevant session data.

**Logging Data:** To facilitate monitoring and debugging, a data structure can be employed to capture logging information. This structure may include timestamps, log levels, error messages, and any contextual data necessary for troubleshooting.

**8.3.2 Global data structure**

## **Captcha Data Structure:** This data structure represents the generated Captcha and its associated attributes. It can include fields such as the Captcha image, solution, complexity level, timestamp, and any other relevant metadata. The Captcha data structure is accessible to components involved in Captcha generation, rendering, validation, and user interaction.

## **User Data Structure:** This data structure contains information related to the user interacting with the Captcha system. It may include fields like the user's ID, IP address, session details, user preferences, and any other relevant user-specific data. The User data structure is shared across components responsible for user management, session handling, and user interaction tracking.

## **Logging Data Structure:** The logging data structure stores logs and relevant information about system events, errors, and activities. It can include fields such as timestamps, log levels, event descriptions, and contextual data. This data structure is accessible to components responsible for logging, error handling, and system monitoring.

## **Configuration Data Structure:** The configuration data structure holds the system configuration settings and parameters. It includes fields such as Captcha length, character set, noise and distortion levels, and other customization options. The Configuration data structure is available to components involved in Captcha generation, rendering, and customization.

## **8.3.3 Temporary data structure**

**Temporary Image Buffer:** During the rendering process, a temporary image buffer can be used to hold the intermediate image data before it is finalized and saved as the Captcha image. This buffer allows for efficient manipulation and modification of the image, such as applying distortion or adding noise, without directly affecting the final Captcha image until all processing steps are completed.

**Temporary String Buffer:** When generating the Captcha solution string, a temporary string buffer can be utilized to build the solution incrementally. This buffer allows for concatenating and manipulating the characters as they are generated or selected, enabling efficient construction of the final solution string.

+----------------------------+

| Random Character |

| Generation |

+----------------------------+

|

v

+---------------------------+

| Image Rendering |

+---------------------------+

|

v

+--------------------------+

| Noise and |

| Distortion |

+--------------------------+

|

v

+-------------------------+

| Validation and |

| Verification |

+-------------------------+

**Figure 8.1: Architecture diagram**

## **8.3.4 Database description**

MySQL Database : Used to store user registration details.

**8.4 Component Design**

## **8.4.1 Class Diagram**

A class diagram is a form of static structural diagram that depicts a system's structure by displaying its classes, characteristics, operations, and object relationships.

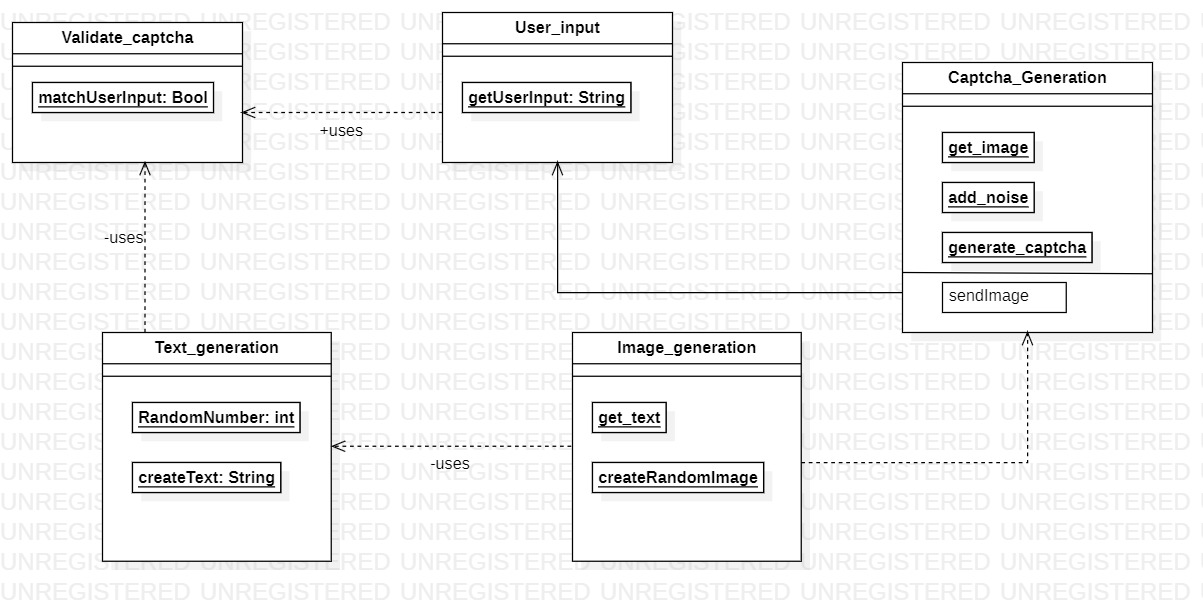


Figure 8.2: Class Diagram

The above diagram shows the relation between all the functions, modules, data structures, their attributes & operations of our project. It also shows “extends” & “aggregation” features. The Captcha\_Generation modules are in aggregation with the user class.

# Project Implementation

**9.1 Introduction**

The implementation phase of the Devanagari CAPTCHA generation project is a crucial step that involves transforming the design specifications into a fully functional system. This phase focuses on coding, testing, and integrating the various components to create a robust and efficient Devanagari CAPTCHA generation system.

During implementation, the development team will utilize the design document and other reference materials to guide the coding process. The team will follow established software development practices, coding standards, and frameworks to ensure high-quality code and maintainable solutions.

**9.2 Tools and Technologies Used**

* Programming Language : Python
* Integrated Development Environment (IDE): Visual Studio
* Image Processing Libraries:
* Web Frameworks: Flask , ReactJs
* Version Control: Git
* Database Management System: MySql
* Documentation and Collaboration Tools

**9.3 Methodologies/Algorithm Details**

**Methodology:**

**Character Selection:** The methodology begins with the selection of Devanagari characters to be included in the CAPTCHAs. This involves determining the desired character set based on the project requirements and the availability of character images.

**Image Generation:** Once the character set is defined, the next step is to generate individual character images. This can be achieved by leveraging image processing techniques and libraries to create visually complex and distorted character representations.

**CAPTCHA Assembly:** The generated character images are then combined to form CAPTCHAs. The assembly process involves randomly selecting a sequence of characters from the character set and arranging them in a visually appealing manner. Additional noise or distortion may be applied to enhance the security and resilience against automated attacks.

**Integration and Deployment:** The final step in the methodology is integrating the CAPTCHA generation system into the target application or platform. This involves incorporating the necessary APIs, libraries, or components to seamlessly integrate CAPTCHA functionality. The system is then deployed to a production environment for real-world usage.

**Algorithm Details:**

**Character Distortion:** To increase the security and prevent automated recognition, various distortion techniques can be applied to the character images. These may include geometric transformations, noise addition, warping, or other image processing operations.

**Background Generation**: The CAPTCHA generation algorithm may include the generation of complex and random backgrounds for the CAPTCHA images. This adds visual complexity and makes it harder for automated algorithms to isolate and recognize individual characters.

**Font and Style Variation:** The algorithm can incorporate font and style variations for each character, making the CAPTCHAs visually diverse and challenging for automated recognition.

**Randomization:** Randomization plays a crucial role in the CAPTCHA generation process. Random selection of characters, their positions, orientations, and other parameters ensures that each CAPTCHA is unique and difficult to crack using automated methods.

**Security Measures:** The algorithm may incorporate additional security measures to protect against attacks, such as rate limiting, IP blocking, or detection of suspicious behavior. These measures help prevent brute-force attacks or malicious activities.

**9.4 Verification and Validation for Acceptance**

# **Software Testing**

**10.1 Type of Testing Used**

**Functionality Testing:** This type of testing focuses on verifying the functional aspects of the Devanagari CAPTCHA system. It ensures that the CAPTCHA is generating and displaying the correct characters, the user input is correctly validated, and the CAPTCHA is integrated properly into the overall system.

**Usability Testing:** Usability testing assesses the user experience when interacting with the Devanagari CAPTCHA. It examines factors such as readability, clarity, and user-friendliness of the CAPTCHA design. Usability testing helps identify any potential issues that may hinder users from accurately interpreting and responding to the CAPTCHA.

**Security Testing:** Security testing aims to evaluate the resilience of the Devanagari CAPTCHA against automated attacks and ensure it effectively prevents unauthorized access. It involves testing for vulnerabilities such as pattern recognition algorithms, OCR bypass techniques, or other methods that may undermine the CAPTCHA's purpose.

**Performance Testing:** Performance testing checks the response time and reliability of the Devanagari CAPTCHA system under different load conditions. This testing helps ensure that the CAPTCHA is generated and verified within an acceptable timeframe, even when the system experiences high volumes of traffic.

**Cross-browser and Cross-platform Testing**: This testing ensures that the Devanagari CAPTCHA functions correctly across different web browsers (such as Chrome, Firefox, Safari, etc.) and various platforms (Windows, Mac, Linux, etc.). It helps verify that the CAPTCHA is consistently displayed and behaves as intended, regardless of the user's browser or operating system.

**Localization Testing:** Since Devanagari is primarily used in languages such as Hindi, Nepali, and Marathi, localization testing ensures that the Devanagari CAPTCHA is culturally appropriate and effectively utilized in the targeted regions. It considers factors like language-specific characters, punctuation, and formatting conventions.

**10.2 Test Cases and Test Results**

**Test Case 1 :** Verify successful login with valid credentials and correct CAPTCHA

**Steps:**

* Enter a valid username and password.
* Enter the correct Devanagari CAPTCHA.
* Click the "Login" button.

**Expected Result:** The user should be successfully logged in.

**Test Case 2 :** Verify error message for invalid username

**Steps:**

* Enter an invalid username.
* Enter a valid password.
* Enter the correct Devanagari CAPTCHA.
* Click the "Login" button.

**Expected Result:** An error message should be displayed indicating that the username is invalid.

**Test Case 3 :** Verify error message for incorrect password

**Steps:**

* Enter a valid username.
* Enter an incorrect password.
* Enter the correct Devanagari CAPTCHA.
* Click the "Login" button.

**Expected Result:** An error message should be displayed indicating that the password is incorrect.

**Test Case 4 :** Verify error message for incorrect CAPTCHA

**Steps:**

* Enter a valid username.
* Enter a valid password.
* Enter an incorrect Devanagari CAPTCHA.
* Click the "Login" button.

**Expected Result:** An error message should be displayed indicating that the CAPTCHA is incorrect.

**Test Case 5:** Verify successful registration with valid credentials and correct CAPTCHA

**Steps:**

* Enter a unique username.
* Enter a valid password.
* Confirm the password.
* Enter the correct Devanagari CAPTCHA.
* Click the "Register" button.

**Expected Result:** The user should be successfully registered.

**Test Case 6 :** Verify error message for an existing username during registration

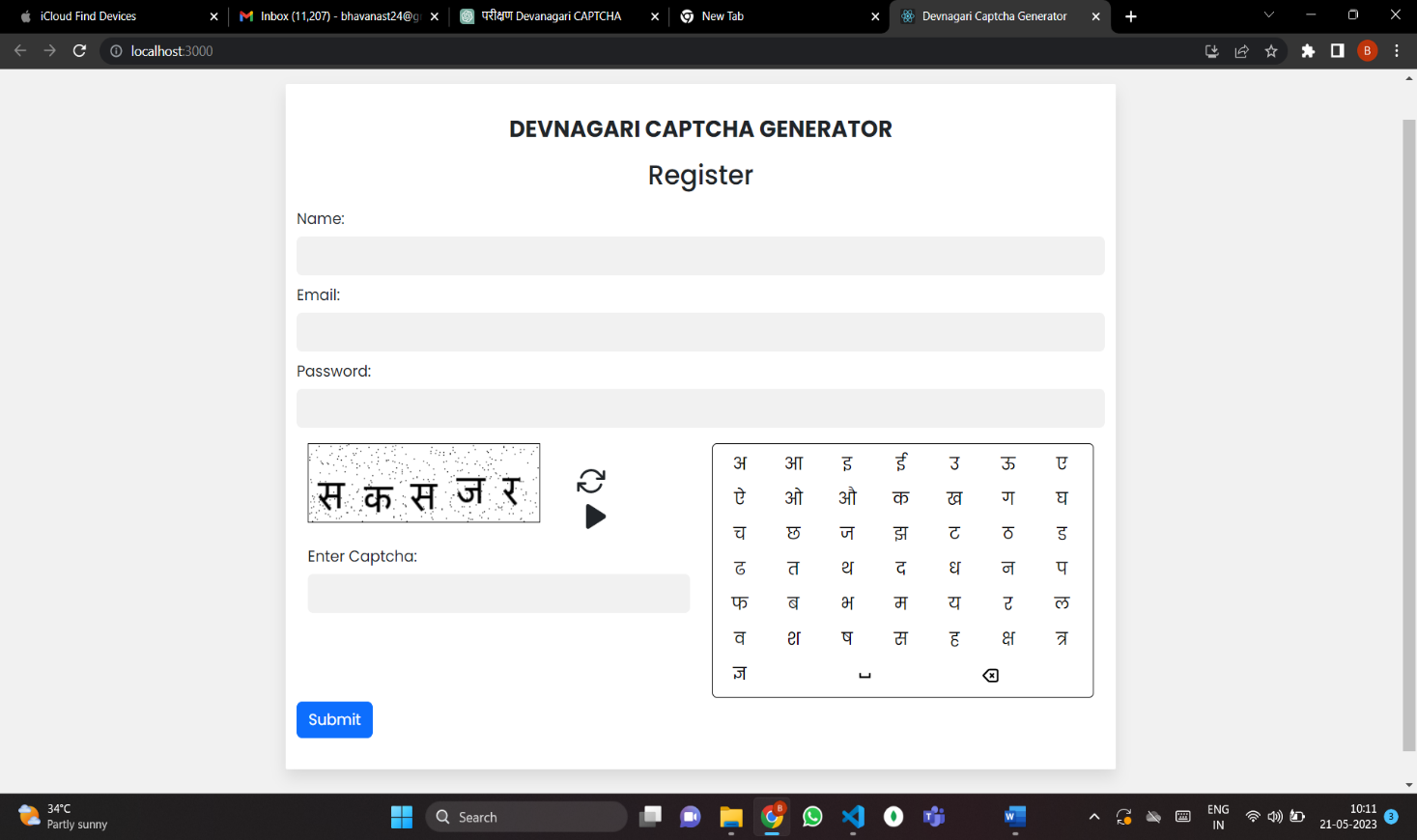
**Steps:**

* Enter an existing username.
* Enter a valid password.
* Confirm the password.
* Enter the correct Devanagari CAPTCHA.
* Click the "Register" button.

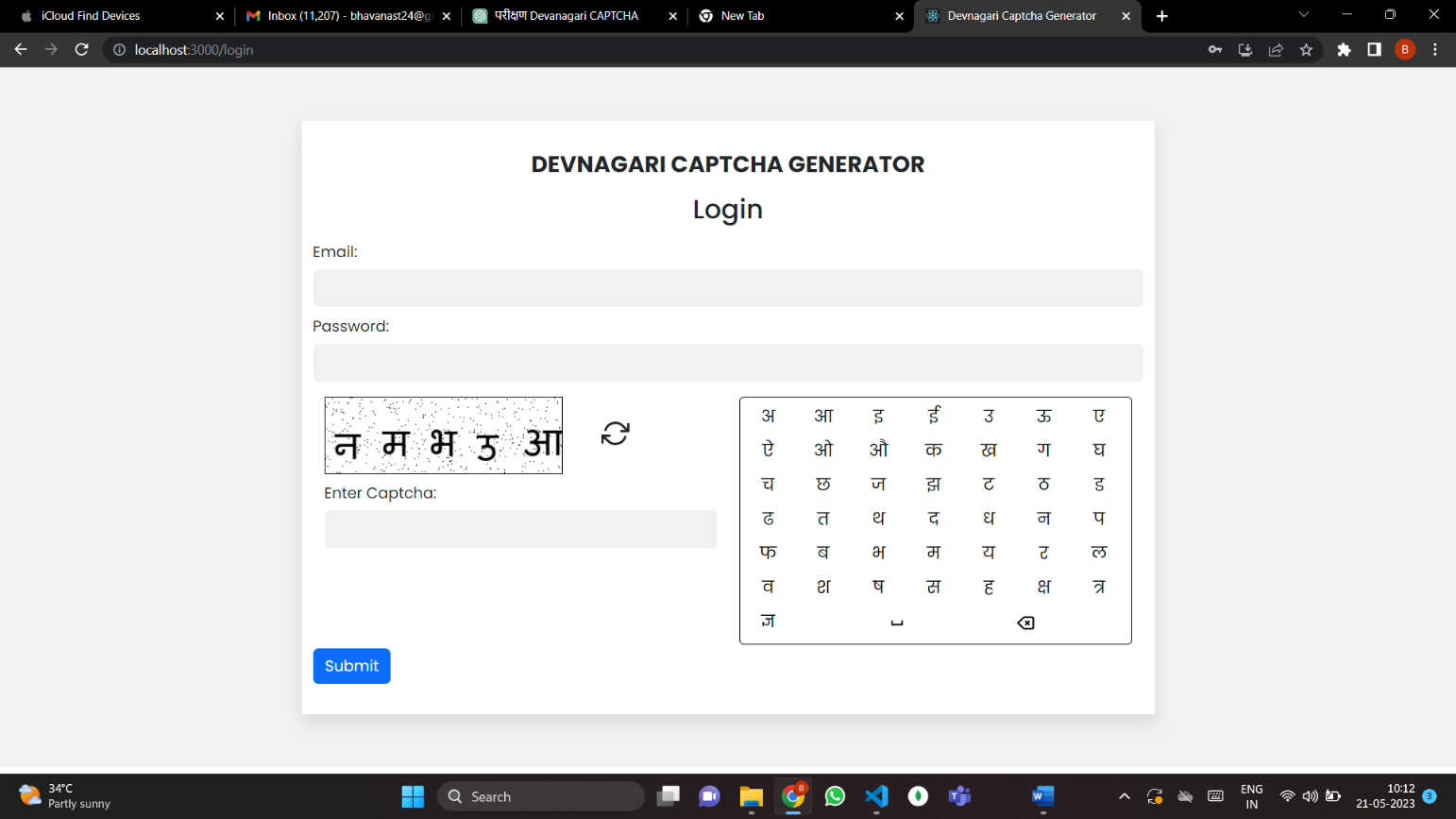
**Expected Result:** An error message should be displayed indicating that the username already exists.

# Results

**11.1 Screenshots**

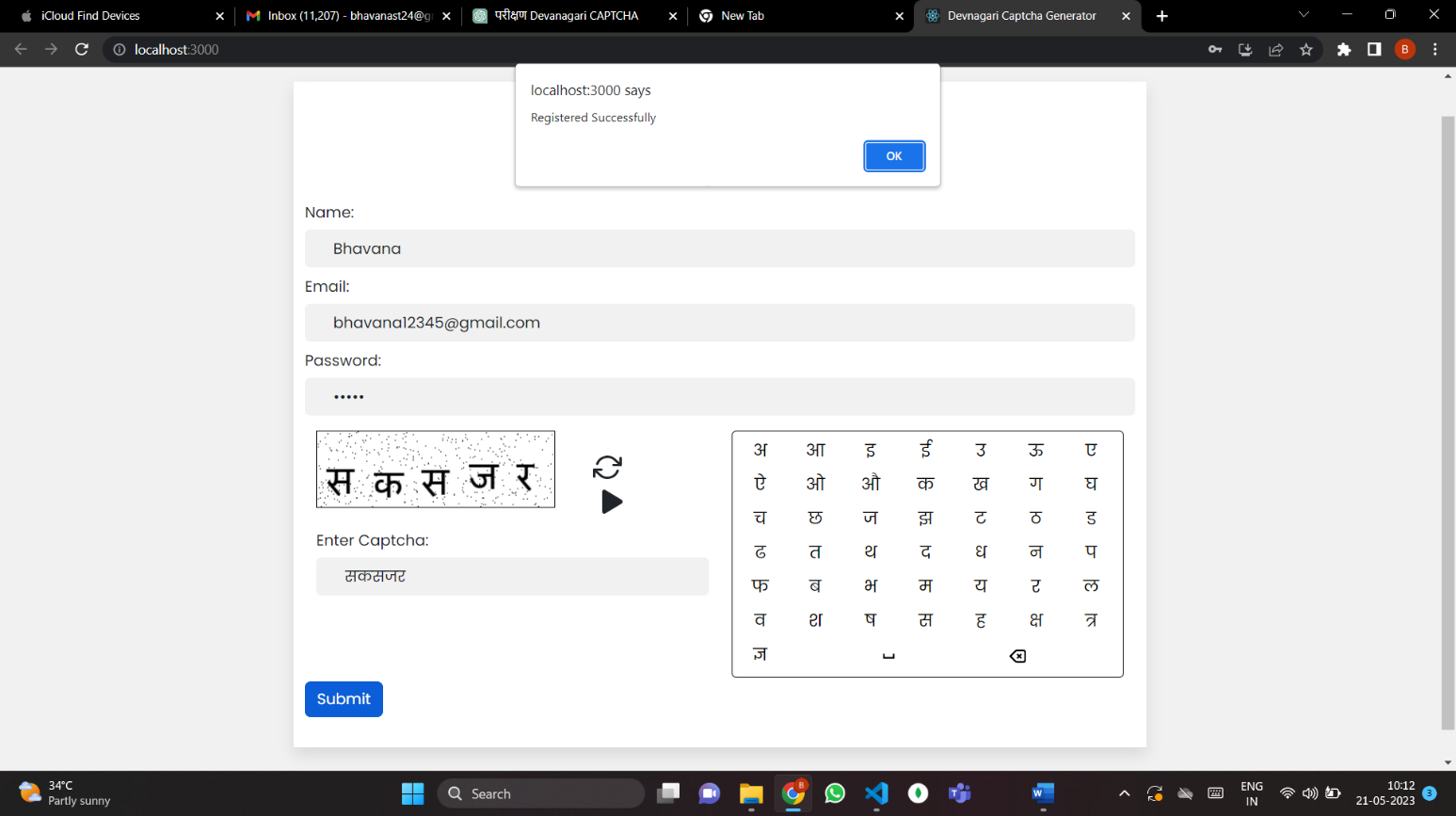


**Fig**  Registration form

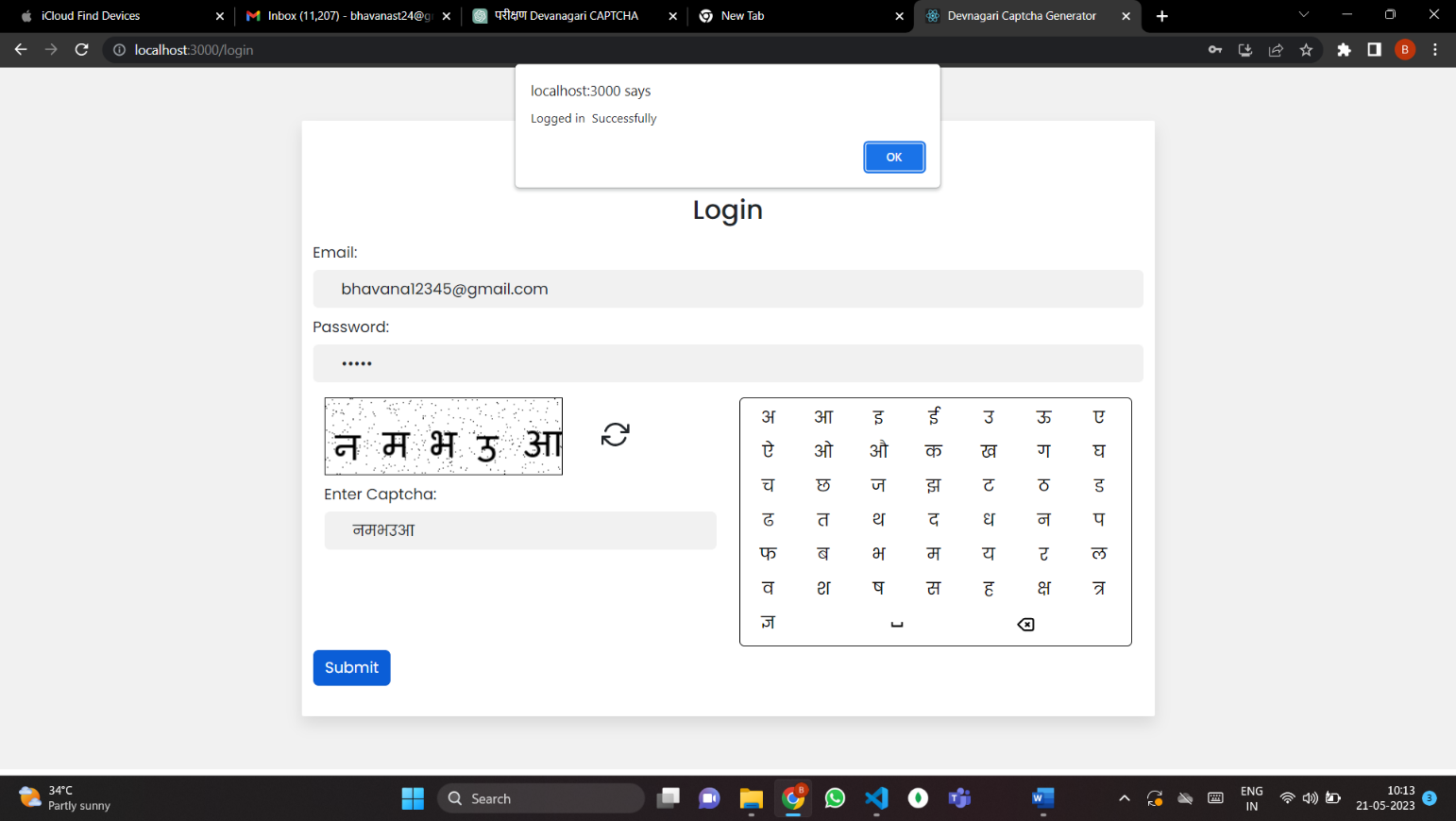


**Fig Login form**

**11.2 Outputs**



**Fig successful Registration**



**Fig successful login**

# **Deployment and Maintenance**

**12.1 Installation and un-installation**

Install the following modules using ‘pip install command’

1. captcha
2. Flask
3. Flask-MySQLdb
4. Flask-SQLAlchemy
5. itsdangerous
6. Jinja2
7. MarkupSafe
8. mysqlclient
9. numpy
10. opencv-python-headless
11. pi
12. python-dotenv
13. SQLAlchemy
14. typing\_extensions
15. Werkzeug
16. zipp

Run the Main.py file from Anaconda Prompt/Terminal/Command Line Prompt

For uninstallation, remove the files from the system and uninstall the above mentioned modules (optional).

**12.2 User help**

* The users will be able to use the application easily which is based on Devanagari Language .
* In case the user has any issues in installation, removal or using the application, he/she may reach out to us on our email-id which will be provided to the customer before they start using the software

**Conclusion and future scope**

**Conclusion :**

Devanagari CAPTCHA is a valuable tool for enhancing security and preventing automated attacks in applications that utilize the Devanagari script. It provides a means of verifying human users by presenting them with a challenge that is typically difficult for automated scripts to solve.

**Future Scope :**

Multilingual CAPTCHA: Expanding the capabilities of Devanagari CAPTCHA to support multiple languages within the same CAPTCHA challenge would be beneficial for applications catering to diverse language requirements.

Mobile-friendly CAPTCHA: With the growing use of mobile devices, optimizing Devanagari CAPTCHA for mobile screens and touch interfaces can improve user experience and ease of interaction.

Adaptive CAPTCHA: Developing adaptive CAPTCHA systems that dynamically adjust the difficulty level based on user behavior and risk analysis can enhance security while minimizing user frustration.

**Bibliography**

[1] Y. Zi, H. Gao, Z. Cheng, and Y. Liu, ”An end-to-end attack on text CAVrCHAs,” IEEE Trans. Inf. Forensics Secur., voL 15, pp. 753-766, 2020.

[2] Y.-W. Chow, W. Susilo, and P. Thomcharoensri, ”CAVTCHA design and security issues,” in Proc. Adv. Cyber Secur.: Principles, Techn., Appl. , 2019, pp. 69-92.

[3] X. Ling et al., ”Deepsec: A uniform platform for security analysis of deep learning model,” in Proc. IEEE Symp. Secur. Privacy, 2019, pp. 673—690.

[4] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, ”Selective learning confusion class for text-based captcha recognition,” IEEE Access, vol. 7, pp. 22246-22259, 2019.

[5] J. Li, S. Ji, T. Du, B. Li, and T. Wang, ”Textbugger. Generating adversarial text against real-world applications,” in Ptoc. Annu. Netw. Distrib. Syst. Secur. Symp., 2019, pp.

**References**

[1] Y. Zi, H. Gao, Z. Cheng, and Y. Liu, ”An end-to-end attack on text CAVrCHAs,” IEEE Trans. Inf. Forensics Secur., voL 15, pp. 753-766, 2020.

[2] Y.-W. Chow, W. Susilo, and P. Thomcharoensri, ”CAVTCHA design and security issues,” in Proc. Adv. Cyber Secur.: Principles, Techn., Appl. , 2019, pp. 69-92.

[3] X. Ling et al., ”Deepsec: A uniform platform for security analysis of deep learning model,” in Proc. IEEE Symp. Secur. Privacy, 2019, pp. 673—690.

[4] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, ”Selective learning confusion class for text-based captcha recognition,” IEEE Access, vol. 7, pp. 22246-22259, 2019.

[5] J. Li, S. Ji, T. Du, B. Li, and T. Wang, ”Textbugger. Generating adversarial text against real-world applications,” in Ptoc. Annu. Netw. Distrib. Syst. Secur. Symp., 2019, pp.

[6] H. Kwon, Y. Kim, H. Yoon, and D. Choi, ’CAPTCHA image generation systems using generative adversarial networks,” IEICE Trans. Inf. Syst., vol. 101, no. 2, pp. 543-546, 2018.

[7] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, ”Selective learning confusion class for text-based captcha recognition,” IEEE Access, vol. 7, pp. 22246-22259, 2019.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, ”Towards deep learning models resistant to adversarial attacks, in Proc. Int. Conf. Learn. Repre- sentations, 2018, pp.

[9] W. Jonathan, Strong CAPTCHA Guidelines: v1.2. http://bitland.net/captcha.pdf, December.

[10] Y. Jeff and A. El Ahmad, “Usability of CAPTCHAs or Usability Issues in CAPTCHA Design,” In SOUPS ’08, pp. 44–52, New York, NY, USA,. ACM.

[11] L. Ying-Lien and H. Chih-Hsiang, “Usability Study Of Text-based CAPTCHAs,” Displays, vol. 32, April. 2011, pp. 81-86.

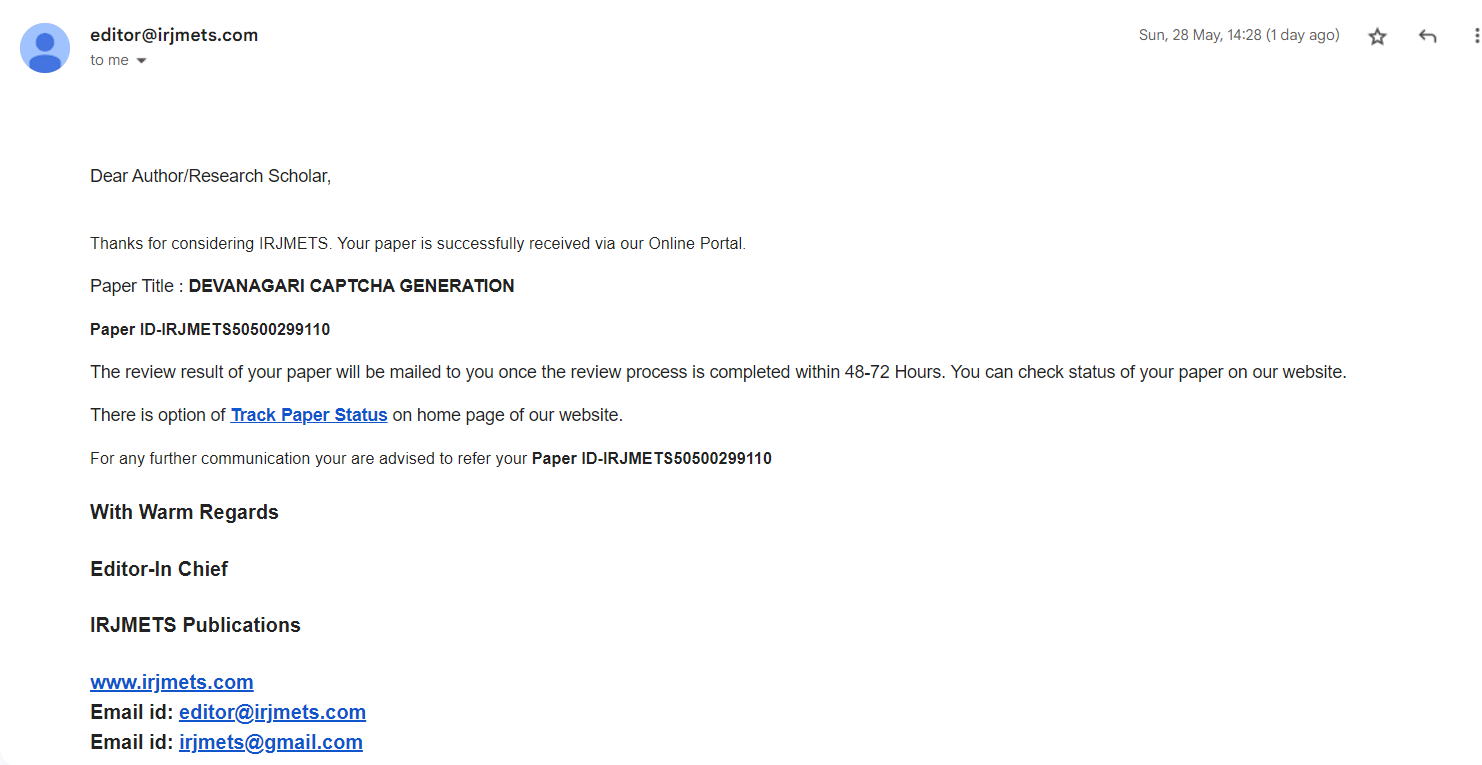
[12] W. Jonathan, Strong CAPTCHA Guidelines: v1.2. http://bitland.net/captcha.pdf, December.

**Project Planner**

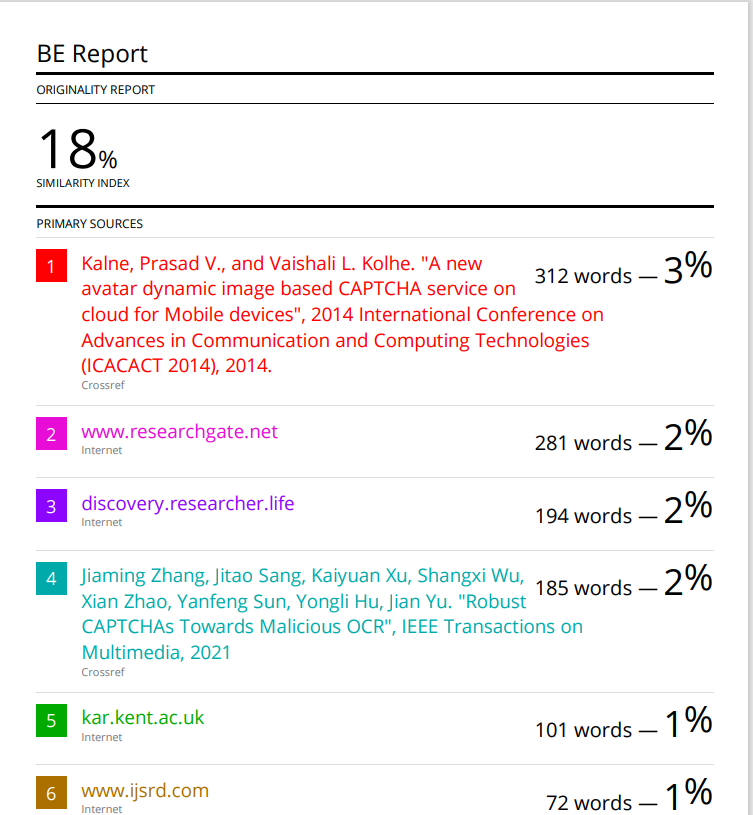
|  |  |  |
| --- | --- | --- |
| **Module** | **Status** | **Flow of Completion** |
| Character Recognition | Completed | August - September |
| CAPTCHA Generation | Completed | September - October |
| Image Creation | Completed | October - November - December |
| Audio Generation | Completed | November - December |
| CAPTCHA Verification | Completed | January - February |
| Database Creation | Completed | February |
| User Interface | Completed | March - April |

# **Appendix Proof of Paper Submitted**

1. Paper Title: Devnagari Captcha Generation
2. Name of the Conference/Journal where paper submitted : IRJMETS (INTERNATIONAL RESEARCH JOURNAL OF MODERNIZATION IN ENGINEERING TECHNOLOGY AND SCIENCE)
3. Paper accepted/rejected : Pending
4. Review comments by reviewer : Pending
5. Corrective actions if any : NA



**Plagiarism Report**



# **Information of Project Group Members**

**MEMBER 1**

1. Name: Bhavana Thakare

2. Date of Birth: 02/04/2001

3. Gender: Female

4. Permanent Address: Lasalgaon, Nashik

5. E-Mail: [bhavanast24@gmail.com](mailto:bhavanast24@gmail.com)

6. Mobile/Contact No: 8805472698

7. Placement Details: Placed at RBL Bank

**MEMBER 2**

1. Name: Mrunal Sadawarte

2. Date of Birth: 17/01/2001

3. Gender: Female

4. Permanent Address: Malkapur, Buldhana

5. E-Mail: [sadawartemrunal@gmail.com](mailto:sadawartemrunal@gmail.com)

6. Mobile/Contact No: 8669885980

7. Placement Details: Placed at RBL Bank

**MEMBER 3**

1. Name : Abhiram Shelkar

2. Date of Birth: 01/09/2001

3. Gender: Male

4. Permanent Address: Akola

5. E-Mail: [abhiramshelkar5@gmail.com](mailto:abhiramshelkar5@gmail.com)

6. Mobile/Contact No : 9561224685

7. Placement Details: Preparing for Masters

**MEMBER 4**

1. Name : Sakshi Ghonge

2. Date of Birth: 30/0/2001

3. Gender: Female

4. Permanent Address: Darwha , Yavatmal

5. E-Mail: sakshighonge3434@gmail.com

6. Mobile/Contact No : 7350673897

7. Placement Details: Not placed.