CHAPTER 1

1.1 Overview Of The Organization

HAEGL TECHNOLOGIES PVT LTD (Formerly known as CulinaryXP) initially incubated at NSRCEL, IIM, Bengaluru and presently functioning at SDM College of Engineering Dharwad aims for the development and implementation of Healthcare, Agriculture and Education technology using Artificial Intelligence platform.

Founders: Gautam Shigaonkar M.Tech [CSE]
Neha R Pudakalakatti M.Tech [CSE]

HAEGL helps differently abled children develop a wide variety of skills and provide them with tools to help them navigate their social and physical world.

HAEGL promotes global farming with the best local solutions ranging from plant protection, crop monitoring and farm management.

HAEGL also aims to create multi-sensory learning aids for children, with a focus on ensuring those with disabilities (visual, developmental, learning) are able to participate equally in learning opportunities.

1.1.1 HAEGL

Shaping Informed Paths: HAEGL, Your Trustworthy Guide in Generative AI Evolution.

Our mission is to assist businesses in making informed decisions when adopting Generative AI, placing emphasis on cautious beginnings rather than prolonged and inefficient decision-making processes. Our team of consultants is ready to address all your concerns and help you overcome typical reservations associated with implementing Generative AI. Talk to our consultants about your concerns related to:

- **Potential Data Leakage:** Recognizing the potential risks of data leakage, we implement robust measures to protect your sensitive information throughout the entire AI integration process.

- **Managing Hallucinations:** Our proficiency centers on mitigating hallucinations that may emerge from Generative AI, guaranteeing the technology delivers precise and dependable outcomes.
- **Ethical And BIAS Considerations:** Dedicated to ethical AI, we diligently reduce biases, fostering fairness and accountability in your systems for integrity and equitability.
- **Integration And Technical Challenges:** Our consultants excel in seamlessly integrating Generative AI into your existing infrastructure, adeptly overcoming technical challenges.

1.1.2 HAEGL Cloud Excellence

What makes HAEGL Cloud 360 the preferred choice

- Data Security Backup
- AI-Driven Reconciliation
- Effortless Data Transfer
- Continuous Upgradation
- Safe and Secure
- Robust and Scalable

1.1.3 Grants Received:

- **2017:** GRANT CHALLENGE KARNATAKA 10 Lakhs.
- **ICAR:**25 Lakhs for SEE & SPRAY ROBOT.
- **Govt of Karnataka[K-TECH]:** 25 Lakhs for Foot Track Balance Rehabilitation for Cerebral Palsy.
- **Children UAS Dharwad:**50 Lakhs for R&D in Agriculture.
- **NRDC:** Shortlisted for 25 Lakhs of Funding.
- **Government of Bihar:** Final Negotiations on Weather Forecasting Analysis using AI & Big Data.
- **DRDO:** Use of Artificial Intelligence in the Defence Sector.

1.2 Vision And Mission of the Organization

1.2.1 HAEGL Vision:

"To emerge as a global leader in Advanced Manufacturing, Research & Development, and Innovation to create an ecosystem for an inclusive, balanced, and sustainable development of the State."

- **Global Leadership:** The vision of HAEGL is to become a prominent figure in three critical areas: Advanced Manufacturing, Research & Development (R&D), and Innovation on a global scale. This suggests a desire to be at the forefront of technological advancements and industrial progress.
- **Ecosystem Creation**: The vision emphasizes the creation of an ecosystem conducive to development. This includes fostering collaboration between industry, academia, government, and other stakeholders to ensure comprehensive and sustainable progress.
- Inclusive, Balanced, and Sustainable Development: The vision underscores the importance of ensuring that development benefits all sections of society, maintains a balance across different sectors, and is sustainable in the long term. This indicates a commitment to social equity, economic stability, and environmental stewardship**.**

1.2.2 HAEGL Mission:

"To be a global manufacturing hub and to achieve sustainable R&D growth through capital infusion, technology transfer, world-class research infrastructure, skill upgradation, and benchmarking of policies and practices to the best global standards."

- **Global Manufacturing Hub:** The mission of HAEGL is to position itself as a central hub for manufacturing activities on a global scale. This involves attracting investment, talent, and resources to establish a robust manufacturing ecosystem.

- **Sustainable R&D Growth:** The mission stresses the importance of sustainable growth in Research & Development. This includes investing in R&D projects, fostering innovation, and ensuring that advancements are made responsibly and with long-term sustainability in mind.

1.2.3 Key Strategies

- **Capital Infusion:** The mission involves injecting capital into various aspects of operations to fuel growth, innovation, and infrastructure development.
- **Technology Transfer:** HAEGL aims to acquire and disseminate cutting-edge technologies from around the world, fostering collaboration and knowledge exchange to stay competitive in the global market.
- **World-Class Research Infrastructure:** The mission emphasizes the importance of establishing state-of-the-art research facilities and capabilities to support high-quality R&D activities.
- **Skill Upgradation:** HAEGL intends to invest in the continuous improvement of workforce skills, ensuring that employees are equipped with the necessary knowledge and expertise to excel in advanced manufacturing and R&D.
- **Benchmarking of Policies and Practices:** The mission involves comparing policies and practices against international standards, ensuring that HAEGL.

1.3 Organization Structure

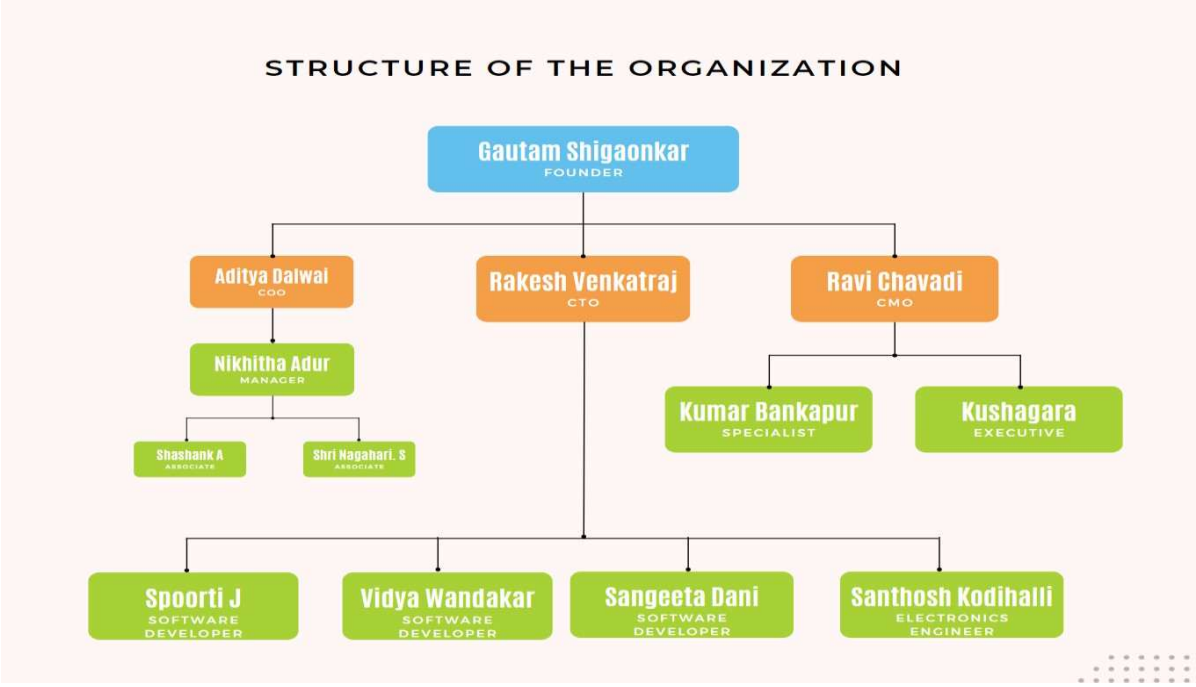


Figure 1.1 Structure of the organization

Figure 1.2. Talent sourcing strategy for the year-2024

1.4 Roles And Responsibilities Of Personnel In The Organization

1.4.1 Founder:

- **The roles and responsibilities of a founder** can vary depending on the nature of the organization, its size, and its stage of development. However, here are some common roles and responsibilities typically associated with founders:
- **Visionary Leadership:** Founders often establish the vision and long-term goals of the organization. They articulate the mission and values that guide the company's activities and decisions.

1.4.2 The Chief Operating Officer (COO) :

Operational Oversight:

- The COO is responsible for overseeing the day-to-day operations of the organization. This involves ensuring that all departments and functions are operating efficiently and effectively to achieve the company's goals and objectives.
- They work closely with department heads to develop operational strategies, streamline processes, and implement best practices to optimize productivity and resource utilization.

1.4.3 The Chief Technology Officer (CTO) :

Technology Strategy Development:

- The CTO is responsible for formulating and executing the organization's technology strategy in alignment with its overall business objectives.

Technology Infrastructure and Architecture:

- The CTO oversees the design, implementation, and maintenance of the organization's technology infrastructure and architecture.

1.4.4 Software Development:

Software Development:

- Design, develop, test, and maintain software applications and systems according to project requirements.

Problem-Solving and Troubleshooting:

- Analyze user requirements and technical specifications to identify potential issues and propose effective solutions.

Software Architecture and Design:

- Architect and design software solutions based on industry best practices and design patterns.

1.4.5 Manager:

Leadership:

- Providing direction, vision, and guidance to the team.
- Motivating team members to achieve individual and collective goals.

Decision-Making*:*

- Making informed decisions based on available data, analysis, and input from team members and stakeholders.

Communication:

- Effectively communicating organizational goals, expectations, and changes to team members.

Associate:

- **Supporting Senior Team Members:** Associates often work closely with more senior team members, providing support in various tasks and projects. This may involve assisting with research, analysis, project management, and other day-to-day activities.

1.4.6 IOT Engineer:

Data Collection and Sensing*:*

- Role: IoT devices are equipped with sensors and actuators to collect data from the physical world.
- Responsibilities: They gather information about environmental conditions, human activities, machine operations, and more. This data is then transmitted to centralized systems for analysis and decision-making.

Data Transmission and Connectivity:

- Role: IoT devices facilitate communication and connectivity between physical objects and digital systems.
- Responsibilities: They transmit collected data over networks such as Wi-Fi, cellular, Bluetooth, or LPWAN (Low Power Wide Area Network) to cloud platforms or other computing systems. Ensuring reliable and secure transmission of data is a crucial responsibility of IoT.

Data Processing and Analytics:

- Role: IoT platforms process and analyze the vast amounts of data collected from connected devices.
- Responsibilities: They employ various techniques such as data filtering, aggregation, pattern recognition, and machine learning algorithms to derive insights and actionable information from raw data. This analysis helps in optimizing processes, predicting failures, and making informed decisions in real-time.

1.5 Products And Market Performance

Software Overview: Hostel Management System for SDM Medicals and UAS Dharwad

Our Hostel Management System is designed to streamline and automate the management of hostel facilities for SDM Medicals and UAS Dharwad. This comprehensive solution offers efficient tools to manage hostel operations, ensuring smooth functioning and enhanced security for resident data.

1.5.1 Key Features:

- **Student Registration and Profile Management:** Easily register students into the system and manage their profiles with relevant information such as personal details, academic information, and hostel preferences.
- **Room Allocation and Vacancy Management:** Facilitate hassle-free room allocation based on predefined criteria and manage room vacancies efficiently.
- **Fee Management:** Automate fee collection and management processes, including generating invoices, tracking payments, and managing outstanding dues.
- **Attendance Tracking:** Monitor student attendance within the hostel premises to ensure safety and security.
- **Complaints and Maintenance:** Allow students to raise complaints or maintenance requests, with a system for tracking the status of these requests until resolution.
- **Visitor Management:** Keep track of visitors entering the hostel premises to enhance security measures.
- **Reporting and Analytics:** Generate insightful reports on various aspects such as occupancy rates, fee collection, attendance trends, and more, enabling informed decision-making.

1.5.2 HAEGL Offerings:

Feature-Rich, All-in-One Cloud ERP Systems for Businesses of All Sizes.

ERP Software

Empower your business with dynamic ERP solutions. Access dependable business insights, utilizing comprehensive reports and advanced reporting systems for seamless operations monitoring.

Revolutionize Your Business Operations with HAEGL Cloud 360°

Embark on a revolutionary journey in business management with HAEGL Cloud 360°. Crafted as the pinnacle offering from HAEGL, Cloud 360° stands as a cutting-edge, second-generation ERP solution poised to transform the landscape of business operations. Whether you helm a sprawling enterprise or a boutique business, our cloud- based ERP solution delivers unparalleled efficiency, intelligence, and user-friendly simplicity right to your fingertips. Welcome to a new era of seamless business management.

Drive Efficiency with HAEGL Integrated ERP Modules

- Business Intelligences
- Customer Management
- Contract Management
- Financial Accounting
- Human Resource Management
- Fixed Assets Management
- Inventory Management
- Project Cost & Budget
- Trading and Distribution

CRM Software

Experience tailored CRM solutions hosted in the cloud, designed with industry-specific features. Launch expansive marketing campaigns, cultivate high-quality leads and surpass service expectations.

HRM Software

Streamline your workforce with an automated HR management software seamlessly integrated with payroll. Effortlessly manage timesheets and attendance for a hassle-free experience.

Generative AI

- Building a Responsible
- Building a Secure
- Building a Sustainable

Future with LLMs and Generative AI

Gartner Says More Than.

80 % Apps seamlessly integrate conversational AI, transforming experiences.

30 % By 2025, AI-infused strategies drive enterprise efficiency and innovation.

1.5.3 HAEGL Pioneering Discoveries

1.5.3.1 HAEGL Hybrid Solar Tunnel Dryer

Our Hybrid Solar Tunnel Dryer, now equipped with a state-of-the-art Thermal Calorific Unit. This groundbreaking technology ensures uninterrupted operation, even during nighttime, providing round-the-clock efficiency.

Figure 1.3 Hybrid Solar Tunnel Dryer

Product Details:

- DRYER CAPACITY: 400 to 2000 sft
- OPERATING TEMPERATURE: 50-70°C
- COVER MATERIAL: Polycarbonate
- CONTROL SYSTEM: RH Temperature Airflow
- MAX VOLTAGE: 240 Volt (V)

Versatile Drying Solutions Tested and Proven

Grapes	**Arecanut**
Initial Moisture: 76% WB	Initial Moisture: 50% WB
Final Moisture: 11% WB	Final Moisture: 14% WB
Chilli	Turmeric
Initial Moisture: 75% WB	Initial Moisture: 80% WB
Final Moisture: 10% WB	Final Moisture: 11% WB

Key Features of The HAEGL Hybrid Solar Tunnel Dryer

- Dual-Energy Source: Harness solar power during the day and seamlessly switch to a thermal calorific unit for continuous operation at night.
- Crop-Specific Optimization: Utilize the integrated IoT system to customize drying parameters based on specific crops, ensuring optimal conditions for diverse agricultural needs.
- Round-the-Clock Efficiency: Enable 24/7 drying capabilities, enhancing productivity and flexibility in various agricultural and industrial applications.
- Eco-Friendly Design: Reduce environmental impact with a sustainable design that maximizes solar energy utilization, contributing to greener and more efficient drying processes.

CHAPTER 2

2.1 Introduction

2.1.1 Cybersecurity

Figure 2.1 Cybersecurity

Cybersecurity is a critical field dedicated to protecting digital systems, networks, and data from unauthorized access, cyber attacks, and malicious activities. As our reliance on digital technology grows, so does the importance of cybersecurity in safeguarding sensitive information, preserving privacy, and ensuring the continuity of business operations. With the proliferation of interconnected devices, cloud computing, and the Internet of Things (IoT), the threat landscape continues to evolve, presenting new challenges and vulnerabilities that require proactive measures and innovative solutions to mitigate. A successful cybersecurity approach has multiple layers of protection spread across the computers, networks, programs, or data that one intends to keep safe. In an organization, the people, processes, and technology must all complement one another to create an effective defense from cyber-attacks. A unified threat management system can automate integrations across select Cisco Security products and accelerate key security operations functions: detection, investigation, and remediation.

Importance of Cybersecurity

In today's connected world, everyone benefits from advanced cyberdefense programs. At an individual level, a cybersecurity attack can result in everything from identity theft, to extortion attempts, to the loss of important data like family photos. Everyone relies on critical infrastructure like power plants, hospitals, and financial service companies. Securing these and other organizations is essential to keeping our society functioning.Everyone also benefits from the work of cyberthreat researchers, like the team of 250 threat researchers at Talos, who investigate new and emerging threats and cyber attack strategies. They reveal new vulnerabilities, educate the public on the importance of cybersecurity, and strengthen open source tools. Their work makes the Internet safer for everyone.

Key aspects of cybersecurity include:

- **Information Security:** Protecting the confidentiality, integrity, and availability of data is fundamental to cybersecurity. This involves implementing measures such as encryption, access controls, and data backups to prevent unauthorized access, tampering, or loss of sensitive information.

- **Network Security:** Securing computer networks against unauthorized access and cyber attacks is critical to maintaining operational continuity. Network security measures include firewalls, intrusion detection systems (IDS), virtual private networks (VPNs), and secure Wi-Fi protocols.

- **Endpoint Security:** Securing individual devices, or endpoints, such as computers, smartphones, and IoT (Internet of Things) devices, is essential for preventing malware infections, data breaches, and unauthorized access. Endpoint security solutions include antivirus software, host-based firewalls, and endpoint detection and response (EDR) systems.

2.2 Advantages of Cyber Security

- **Protection of Sensitive Information:** Cybersecurity measures help safeguard sensitive data such as personal information, financial records, and intellectual property from unauthorized access and exploitation by malicious actors.
- **Prevention of Cyber Attacks:** Effective cybersecurity measures help prevent cyber attacks such as malware infections, phishing scams, and denial-of-service (DoS) attacks, reducing the risk of disruptions to business operations and financial losses.
- **Preservation of Privacy:** By implementing robust cybersecurity controls, individuals and organizations can maintain their privacy and confidentiality, ensuring that personal and sensitive information remains protected from unauthorized disclosure.

2.3 Disadvantages of Cyber Security

- **Cost and Resource Intensive:** Implementing and maintaining effective cybersecurity measures can be costly and resource-intensive, requiring investments in technology, personnel, training, and ongoing monitoring and maintenance.
- **Complexity and Technical Challenges:** Cybersecurity involves navigating complex technical challenges, including evolving threats, diverse attack vectors, and rapidly changing technologies, which can pose challenges for organizations with limited expertise and resources.
- **Potential Impact on Usability:** Stringent cybersecurity measures, such as multi-factor authentication and encryption, may introduce usability challenges for users, leading to friction in accessing digital services and systems.

2.4 Different types of attacks in Cybersecurity

- **SQL Injection**
- **Broken Authentication**
- **Brute Force**

2.4.1 Overview of Attacks:

2.4.1.1 SQL Injection:

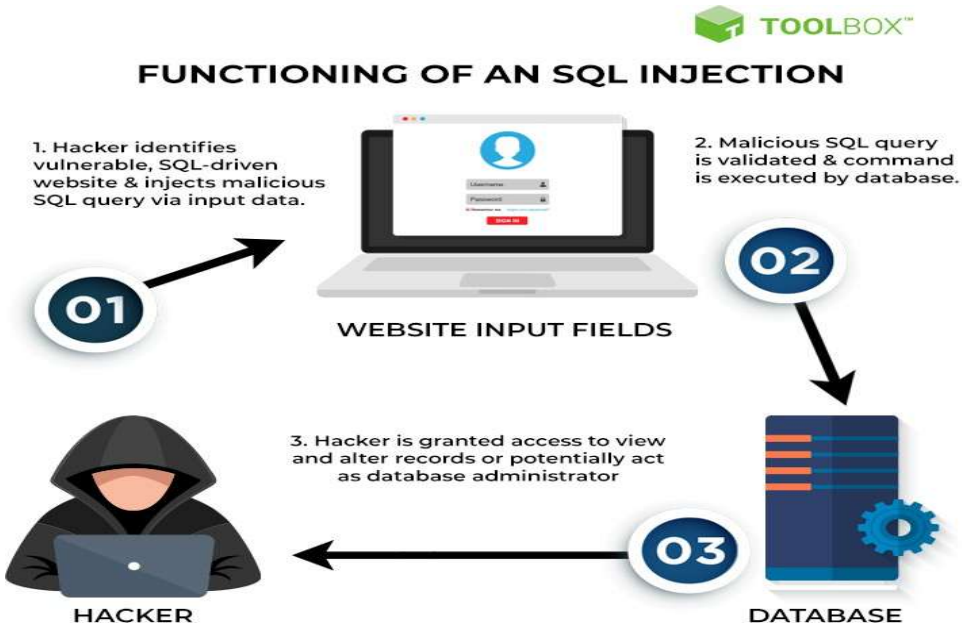


Figure 2.2. Function of Sql Injection

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

SQL Injection Attack Performed

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker

can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.

You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.

2.4.1.2 Broken Authentication

Broken Authentication is a critical security vulnerability that occurs when an application's authentication mechanisms are improperly implemented, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities. It is listed as the second most critical vulnerability in the OWASP Top Ten Web Application Security Risks.

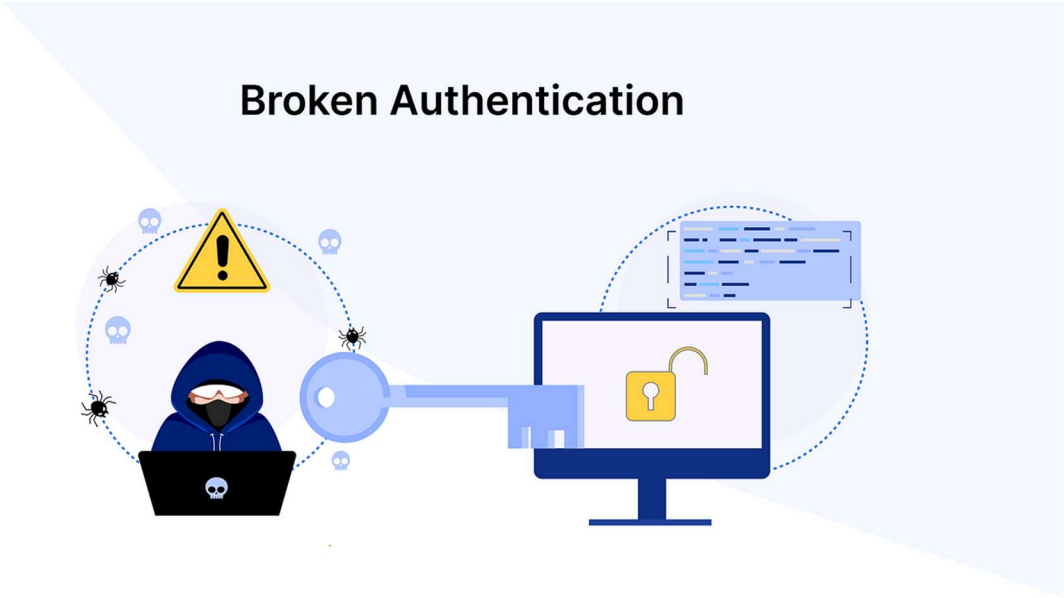


Figure 2.3. Function of Broken Authentication

Impact of a Broken Authentication:

Broken authentication occurs when applications fail to properly protect user credentials and session tokens. This can allow attackers to gain unauthorized access to accounts, including admin accounts, leading to data breaches, identity theft, financial loss, and reputational damage. It may also result in platform abuse and legal issues due to non-compliance with data protection laws.

2.4.1.3 Brute Force

Figure 2.4. Brute Force

A brute force attack is a hacking method where an attacker systematically tries all possible combinations of passwords, encryption keys, or login credentials until they find the correct one. This technique relies on computational power and persistence rather than exploiting software vulnerabilities. There are several types of brute force attacks, including simple brute force attacks, which try every possible combination; dictionary attacks, which use a list of commonly used passwords; and hybrid attacks, which combine dictionary words with numbers or symbols. Other variations include credential stuffing, where attackers use leaked username-password pairs, and reverse brute force attacks, which apply a common password to multiple usernames. To prevent brute force attacks, users and organizations should implement strong, complex passwords, enable multi-factor authentication (MFA), limit login attempts, use CAPTCHA to block bots, and monitor authentication logs for unusual activity. These measures help strengthen security and reduce the risk of unauthorized access.

Impact of Brute Force Attacks

- **Unauthorized Access** – Attackers can gain control over accounts, systems, or databases.
- **Data Breaches** – Sensitive personal or business data can be stolen, leading to privacy violations.
- **Financial Losses** – Stolen banking credentials or fraud can result in direct financial damage.
- **Identity Theft & Fraud** – Compromised personal data can be used for fraudulent activities.
- **System Downtime** – Continuous login attempts can overwhelm servers, causing service disruptions.
- **Reputational Damage** – Businesses can lose customer trust and credibility.
- **Legal & Regulatory Consequences** – Companies may face fines or legal actions for failing to protect user data.
- **Credential Stuffing Risks** – If passwords are reused, multiple accounts can be compromised across platforms.

Examples of Brute Force Attacks

1. **Attack on Yahoo (2012-2016)** – One of the largest data breaches in history, where attackers used credential stuffing (a type of brute force attack) to gain access to user accounts, affecting billions of users.
2. **Mirai Botnet (2016)** – The Mirai malware used brute force attacks on IoT devices with weak or default passwords, turning them into a massive botnet that launched distributed denial-of-service (DDoS) attacks.
3. **Facebook Hack (2019)** – Attackers used brute force techniques to exploit vulnerabilities in Facebook's system, compromising millions of user accounts.
4. **Tesla Cloud Attack (2018)** – Hackers gained access to Tesla's cloud environment through brute force methods, allowing them to mine cryptocurrency using Tesla's computing resources.
5. **WordPress Website Attacks** – Hackers often target WordPress sites using automated brute force tools to guess admin login credentials, leading to website defacement or data theft.

2.5 Literature Survey

2.5.1 SQL Injection:

Scenario 1:

If we use Django authentication system and Encryption techniques.
Here we are using Django Encryption and Authentication technique Where, the passwords are encrypted using hashing algorithms and used the in-built authentication systems to authenticate the original users.

In Database

As you can see here, the password is fully encrypted.
For an instance, if attacker try's to perform SQL injection on the login page, It is very hard to perform and the success rate is very low (possibility of attack).

id	password	last_login	is_superuser	username	email	is_staff	is_active
Fil...	Filter	Filter	Filter	Filter	Filter	Filter	Filter
3	pbkdf2_sha256$870000$CzMP46abWEGMFTq…	2025-03-27 06:05:45.798872	0	doctor	doctor@gmail.com	0	1
4	pbkdf2_sha256$870000$FOCzLxX5AkLuimy…	2025-03-19 16:11:05.756208	0	patient2@gmail.com	patient2@gmail.com	0	1
6	pbkdf2_sha256$870000$jQyJLAwxw42PH3q…	2025-03-27 06:11:05.721164	0	manjupa@gmail.com	manjupa@gmail.com	0	1
8	pbkdf2_sha256$870000$fMU9lBl87C1uHop…	2025-03-19 16:09:33.001708	0	manjunh	manju@gmail.com	0	0
9	pbkdf2_sha256$870000$xfwvYdXwPH7QMyR…	2025-03-20 09:07:16.549689	0	ishwar	ishwar@gmail.com	0	1
14	pbkdf2_sha256$870000$rLHrQ17hOYBbmm4…	2025-03-27 06:13:24.269068	0	hareesh	hareesh@gmail.com	0	1
15	pbkdf2_sha256$870000$ImY4VR32AFPj7FW…	2025-03-27 06:03:12.767471	0	manoj	manoj156@gmail.com	0	1

Scenario 2:

If we not use encryption techniques while registering the new user, it is vulnerable and easy to exploit like (SQL Injection)

For Example:

Note: Here we are not using encryption technique.

In Database:

8	admin299	2024-03-19 07:03:39.816074	0	admin3
9	pbkdf2_sha256$720000$TEVL2fOgRn...	2024-03-19 07:04:55.921868	0	admin4
10	pratham2004	2024-03-19 07:16:31.197640	0	pratham2

As you can see here, in the last user the password in not encrypted. So here, an attacker try's to write a **sql query** in the login's source code, such that it checks for the user in the database. If the password is not encrypted it can be easily exploited by an attacker, which lead to data breach, data integrity failure, etc

Working:

1. At first give any name in login page, which possibly exist in database.
2. At the password column, write a malicious SQL injection code.

   Ex: " OR 1=1 "
3. If the user's password in not encrypted, it directly redirects to main patients dashboard.
4. Here attacker is successful in performing SQL injection attack.

2.5.2 Broken Authentication

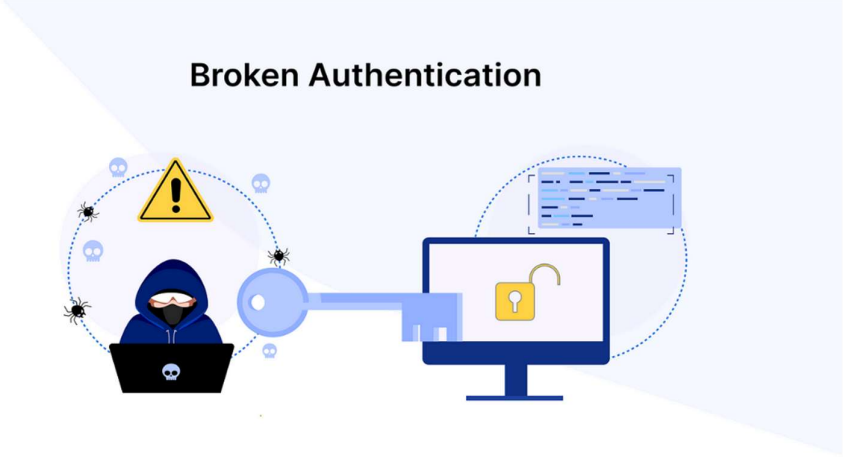


Figure 2.5 Function of Broken Authentication

Attacking Steps for Broken Authentication:

The steps involved in a CSRF attack are as follows:

1.**Credential Stuffing**
- Use stolen username/password combos from previous breaches.

2.**Brute Force Attack**
- Try many passwords on a single account until one works.

3.**Password Spraying**
- Use a few common passwords across multiple usernames.

4.**Session Hijacking**
- Steal and reuse session cookies to impersonate users.

5.**Session Fixation**
- Set a known session ID before login and reuse it after user logs in.

6.**Bypassing 2FA/MFA**
- Exploit weak/misconfigured 2FA via phishing or SIM swaps.

7.**User Enumeration**
- Analyze system responses to discover valid usernames.

Scenario 1:

Broken Authentication: Detection and Mitigation Strategies" by John Smith et al. (2015):

- In this paper, Smith et al. provide a comprehensive analysis of Broken Authentication vulnerabilities and propose a multi-layered approach to detect and mitigate these issues. The authors emphasize the importance of behavioural analysis and anomaly detection to identify suspicious login patterns, such as brute force attacks or credential stuffing. They also introduce a dynamic authentication framework that adapts security measures based on risk levels, such as requiring multi-factor authentication (MFA) for high-risk login attempts.

Scenario 2:

Machine Learning for Detecting Broken Authentication Attacks" by Rajesh Kumar et al. (2020):

Kumar et al. explore the use of machine learning (ML) techniques to detect and prevent Broken Authentication attacks. The authors propose an ML-based model that analyzes login attempts and identifies patterns associated with brute force attacks, credential stuffing, and session hijacking. The model is trained on large datasets of legitimate and malicious login attempts, enabling it to detect anomalies with high accuracy.

2.5.3 Brute Force

A brute force attack is a hacking method where attackers try all possible password or login combinations to gain access. Variants include simple brute force, dictionary attacks (common passwords), hybrid attacks (words + symbols), credential stuffing (leaked credentials), and reverse brute force (one password on many accounts). To prevent these, use strong passwords, enable multi-factor authentication, limit login attempts, add CAPTCHA, and monitor login activity.

Impact of a Successful Attack:

- **Unauthorized Access:** Attackers can infiltrate accounts, systems, or networks.
- **Data Theft:** Sensitive information like personal data and financial records can be stolen.
- **Account Takeover:** Attackers gain full control of user accounts.
- **System Compromise:** Malware installation and operational disruptions may occur.
- **Reputation Damage:** Breaches can erode trust and lead to financial loss.

Examples of Brute Force Attacks

- **Password Cracking:** Automated tools systematically guess passwords.
- **Encryption Key Cracking:** Attackers try all encryption keys to decrypt files.

Scenario 1:

Brute Force Attacks: Techniques and Defenses" (2008) – Discusses attack methods and defenses.

Brute Force Attacks: Techniques and Defenses" (2008) provides an overview of brute force attack strategies and ways to defend against them. It explains how attackers use methods like simple password guessing, dictionary attacks (using common password lists), and hybrid attacks that mix dictionary words with symbols or numbers. Advanced techniques such as credential stuffing and distributed brute force attacks, where multiple systems are used to bypass rate limits, are also covered. On the defensive side, the document discusses implementing strong password policies, account lockout mechanisms, multi-factor authentication (MFA), and rate limiting as effective ways to mitigate brute force threats.

Scenario 2:

A Comprehensive Review of Brute Force Attacks (2015):

Provides an in-depth analysis of how brute force attacks are carried out, their consequences, and how they can be prevented. The review outlines various attack vectors, including direct password guessing, dictionary-based attacks, and distributed brute force via botnets. It highlights the impacts such as unauthorized access, data breaches, financial losses, and reputational damage. The paper also emphasizes prevention techniques like enforcing strong password policies, implementing account lockout mechanisms, using multi-factor authentication, employing intrusion detection systems, and applying proper encryption methods to reduce vulnerability to these attacks.

2.6 System Specification

2.6.1 Hardware Specification:

- Processor - Intel Core i5: An Intel Core i5 processor offers robust performance and security capabilities, ideal for software development and cybersecurity tasks like coding, compiling, and malware analysis.

- Memory (RAM) - 8 GB: With 8 GB of RAM, strike a balance between performance and affordability, suitable for everyday tasks. Consider upgrading for memory-intensive activities like video editing or running multiple demanding applications simultaneously.

2.6.2 Software Specification:

- Operating System – Windows 10, 11: Choose between Windows 10 or 11, both renowned for user-friendly interfaces, robust security, and software compatibility, ensuring a modern computing experience.

- Web Browser – Google Chrome, Microsoft Edge, Brave: Select from Google Chrome, Microsoft Edge, or Brave for fast, secure, and customizable web browsing, each offering unique features and extensions to enhance productivity and privacy.

2.7 Development Tools Used

2.7.1 Frontend :

HTML provides the structural foundation of web pages, while CSS styles and presents elements, ensuring aesthetic appeal and responsiveness. JavaScript adds interactivity and dynamic behavior, enhancing user experience and enabling full- stack development.features like flexbox and grid layout offer powerful tools for creating complex and responsive layouts efficiently. Additionally, preprocessors like Sass and LESS extend CSS capabilities by introducing variables, mixins, and other programming constructs, streamlining the development process.

2.7.2 Backend:

Django, a Python framework, simplifies backend development with its MVC architecture and built-in features like ORM and authentication. It ensures scalability, security, and rapid development of complex web applications.

2.7.3 Database:

SQLite offers a lightweight, serverless, and self-contained solution for small to medium-scale applications, requiring minimal setup and administration. Its SQL support and easy integration make it ideal for various use cases, including mobile apps and embedded systems.

2.8. Design

2.8.1 ER-diagram For Clinic Management System

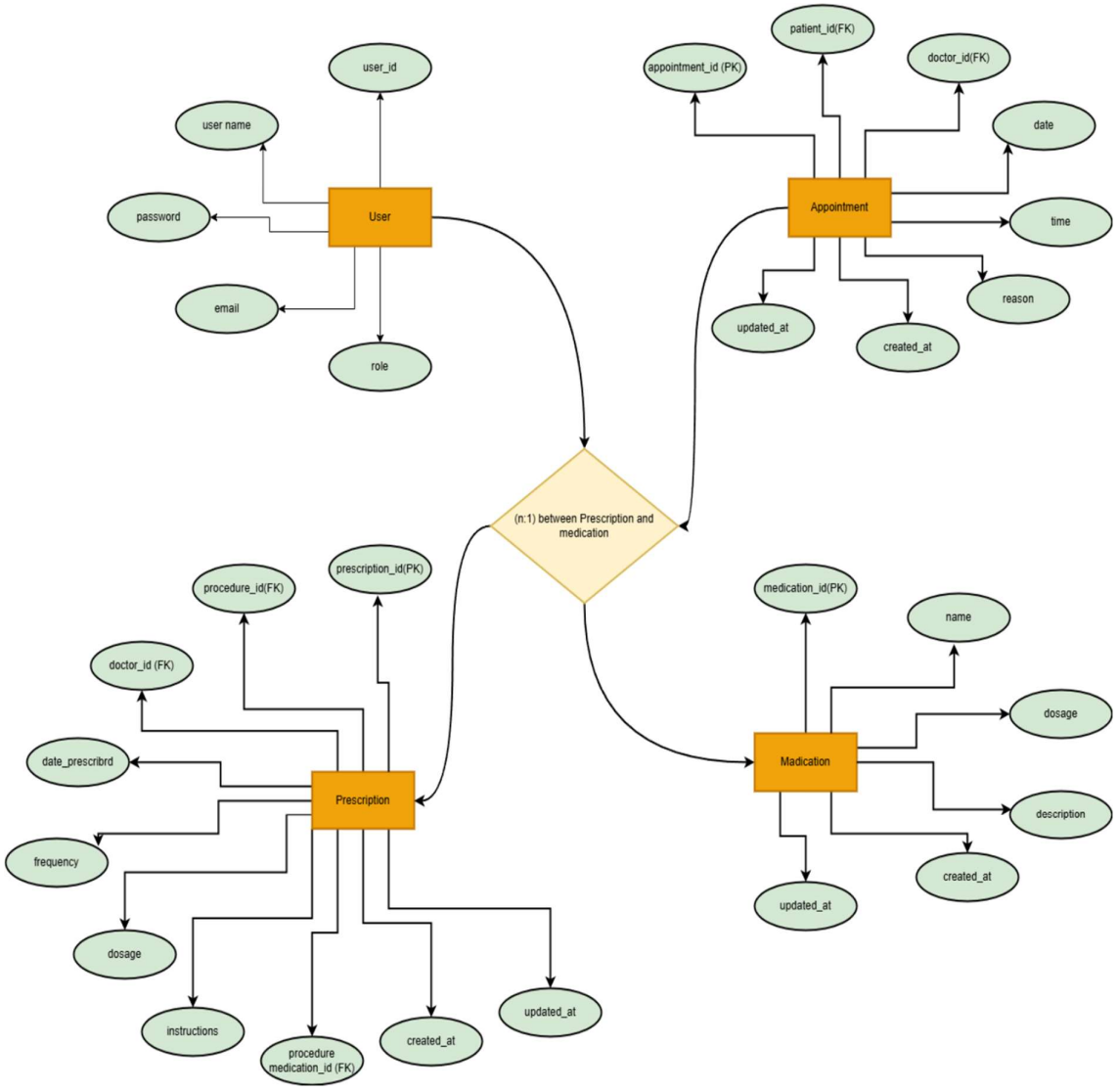
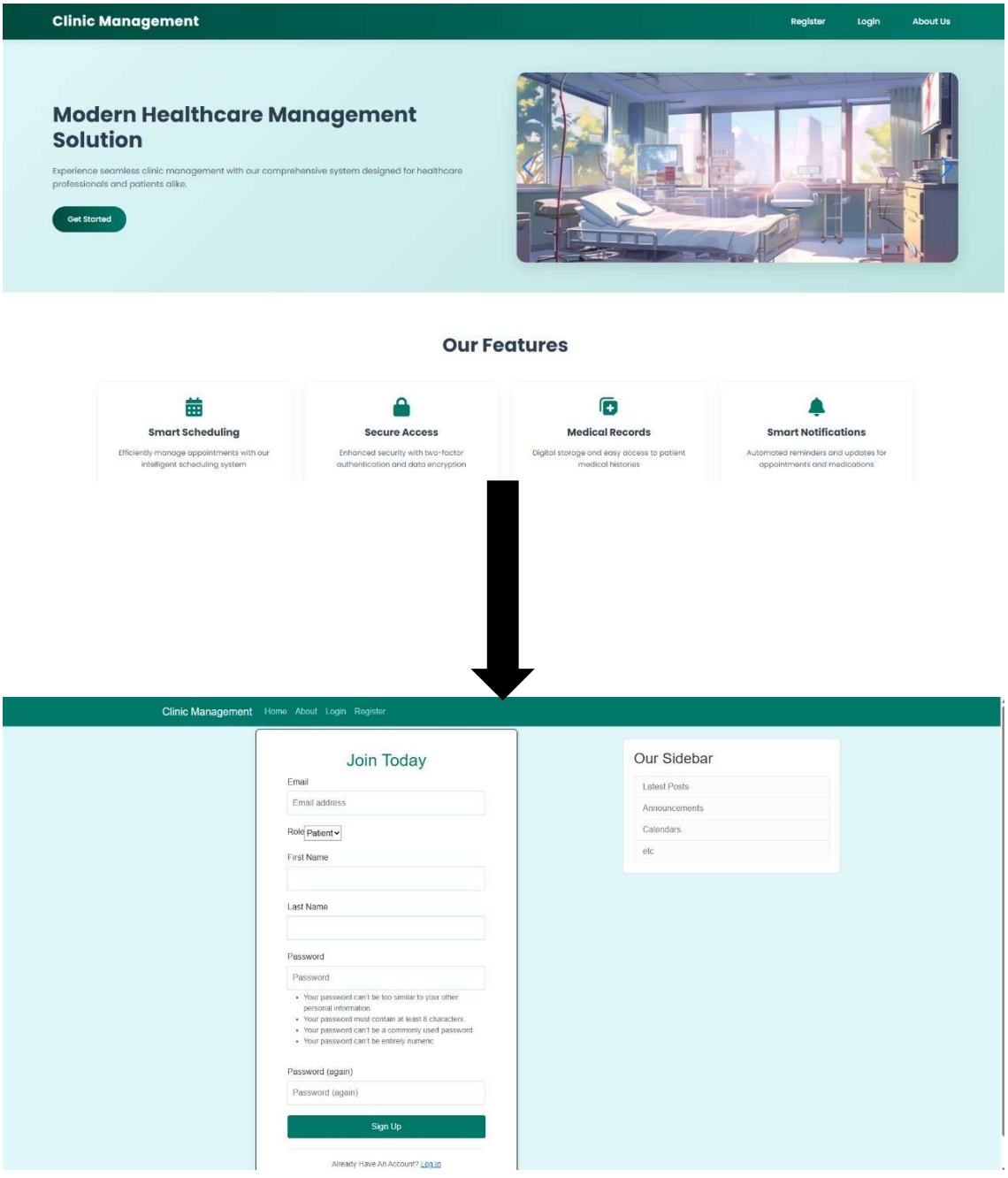
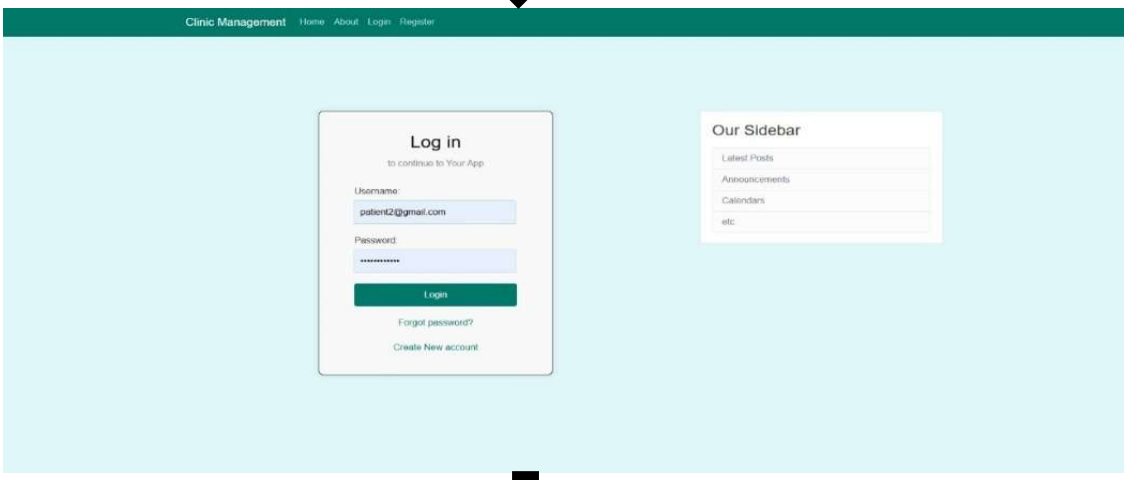
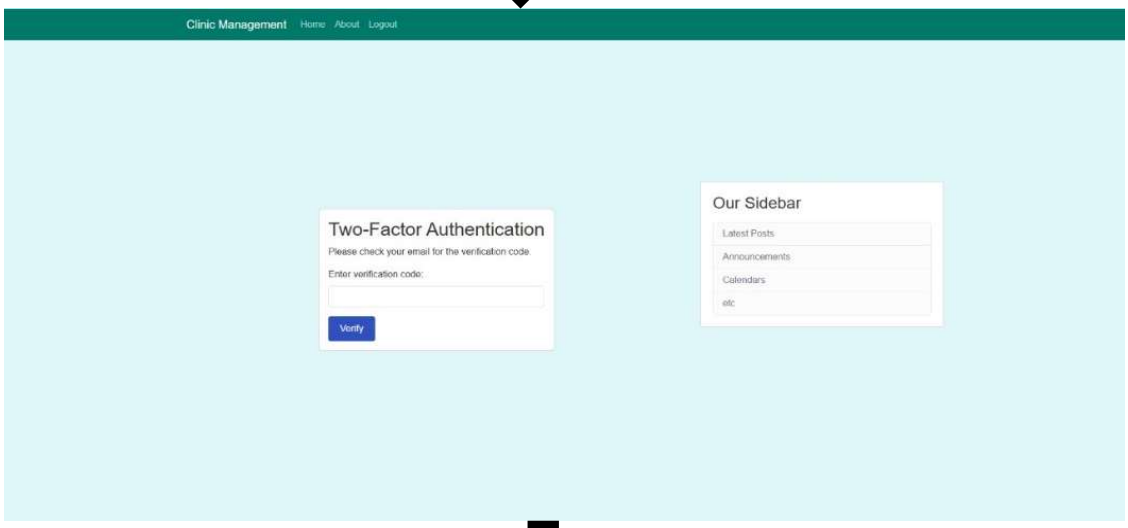


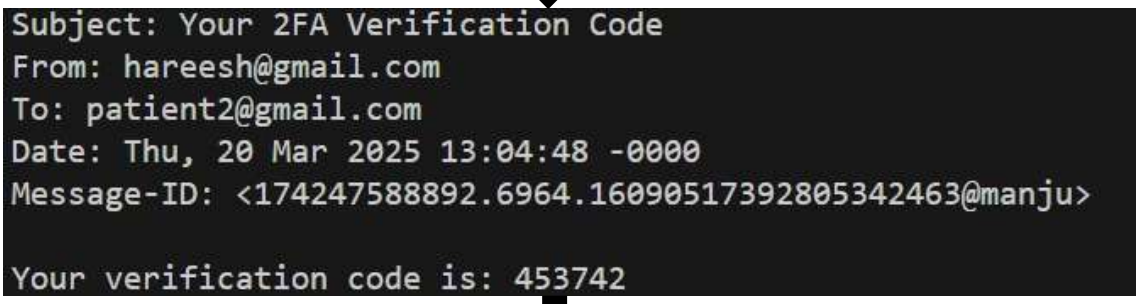
Figure 2.8.1 ER-diagram For Clinic Management System

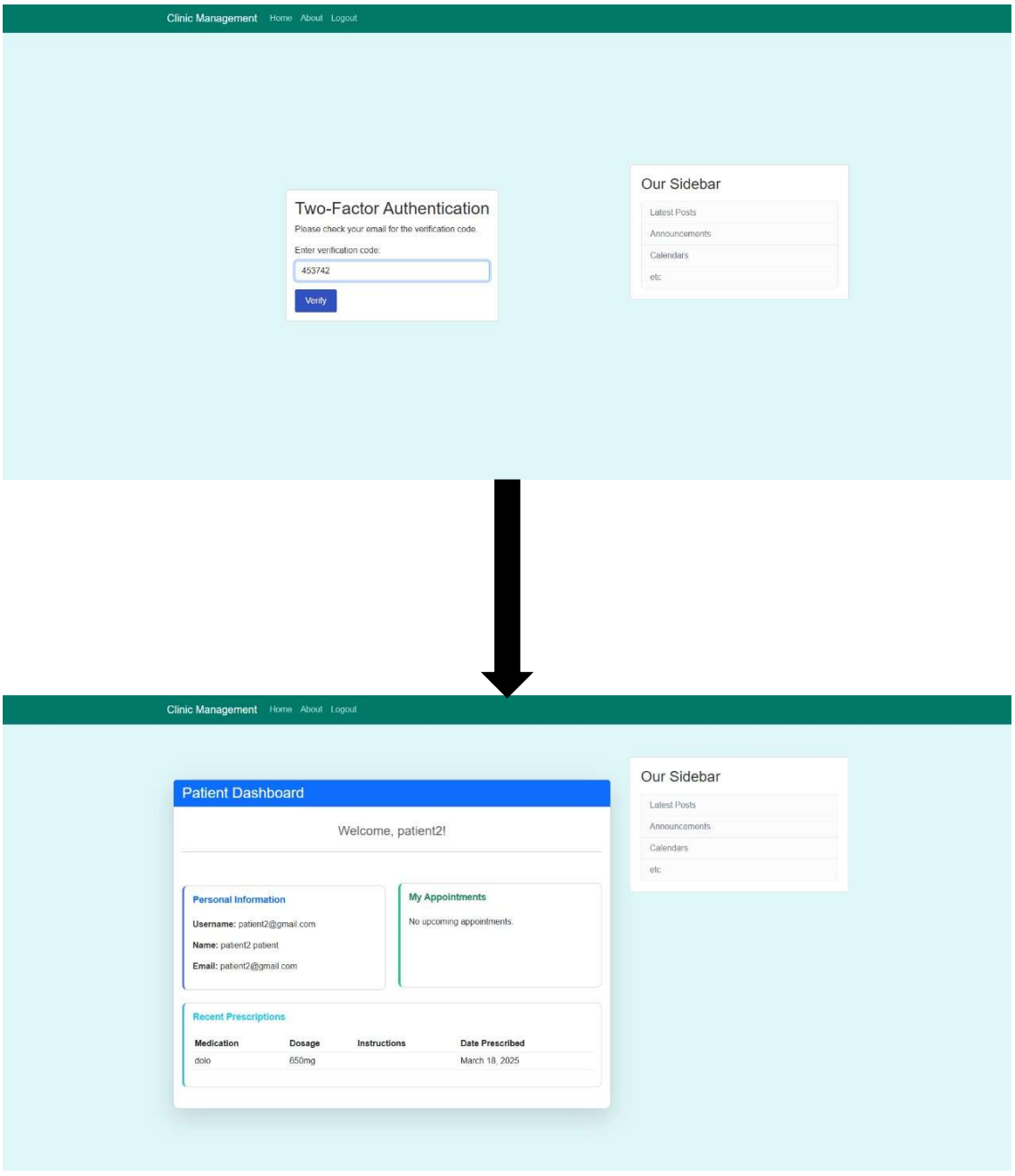
2.8.2 Architectural Diagram of Website




Clinic Management
Home
About
Login
Register
Log in
to continue to Your App
Username:
patient2@gmail.com
Password:
Login
Forgot password?
Create New account
Our Sidebar
Latest Posts
Announcements
Calendars
etc


Clinic Management
Home
About
Logout
Two-Factor Authentication
Please check your email for the verification code
Enter verification code:
Verify
Our Sidebar
Latest Posts
Announcements
Calendars
etc


Subject: Your 2FA Verification Code
From: hareesh@gmail.com
To: patient2@gmail.com
Date: Thu, 20 Mar 2025 13:04:48 -0000
Message-ID: <174247588892.6964.16090517392805342463@manju>
Your verification code is: 453742


Clinic Management
Home
About
Logout
Two-Factor Authentication
Please check your email for the verification code.
Enter verification code:
453742
Verify
Our Sidebar
Latest Posts
Announcements
Calendars
etc
Clinic Management
Home
About
Logout
Patient Dashboard
Welcome, patient2!
Personal Information
Username: patient2@gmail.com
Name: patient2 patient
Email: patient2@gmail.com
My Appointments
No upcoming appointments.
Recent Prescriptions
Medication
Dosage
Instructions
Date Prescribed
dolo
650mg
March 18, 2025
Our Sidebar
Latest Posts
Announcements
Calendars
etc

2.8.3 Class Diagram

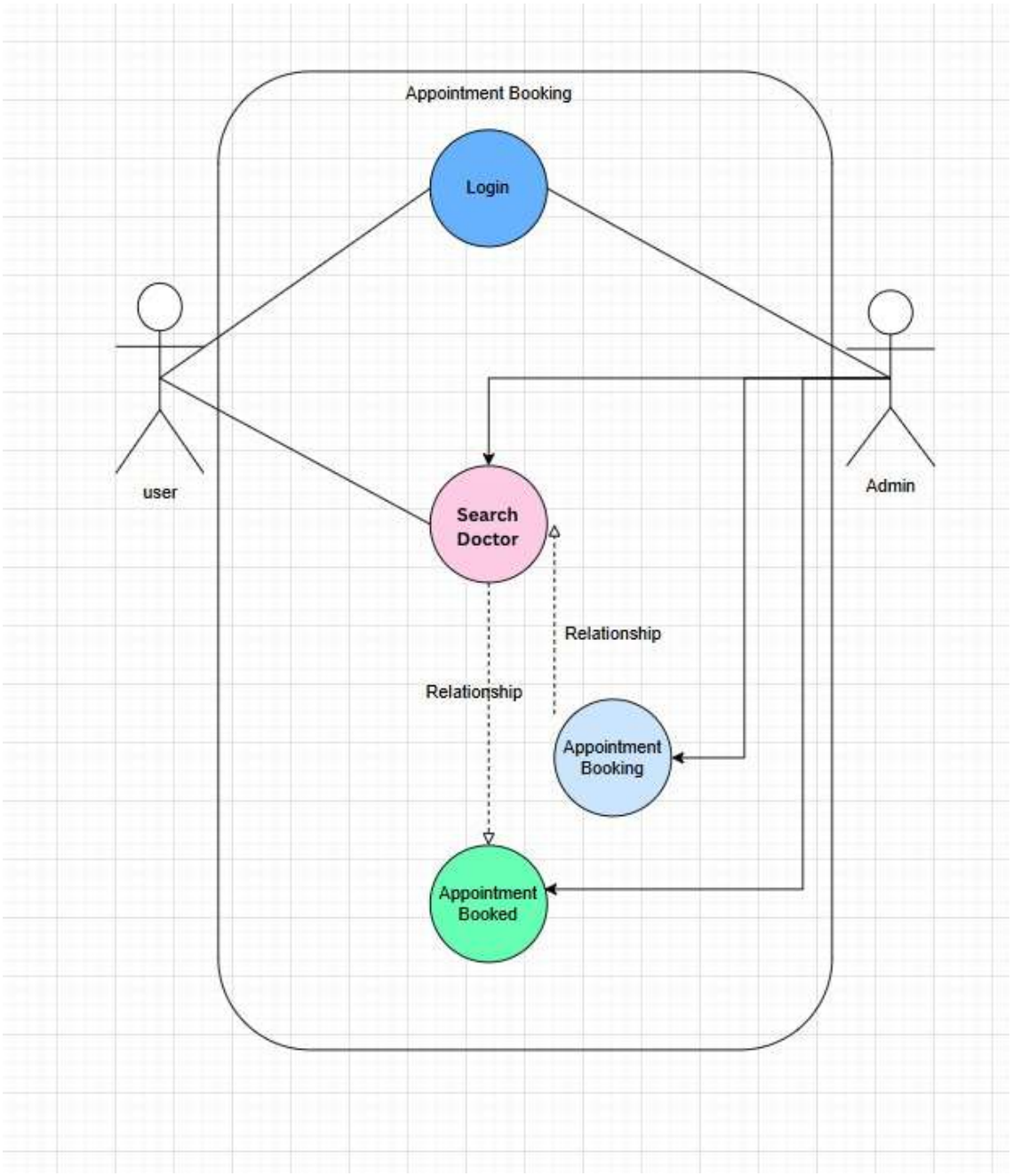


Figure 2.8.3 Class Diagram

CHAPTER 3

3.1 Implementation

3.1.1 SQL Injection:

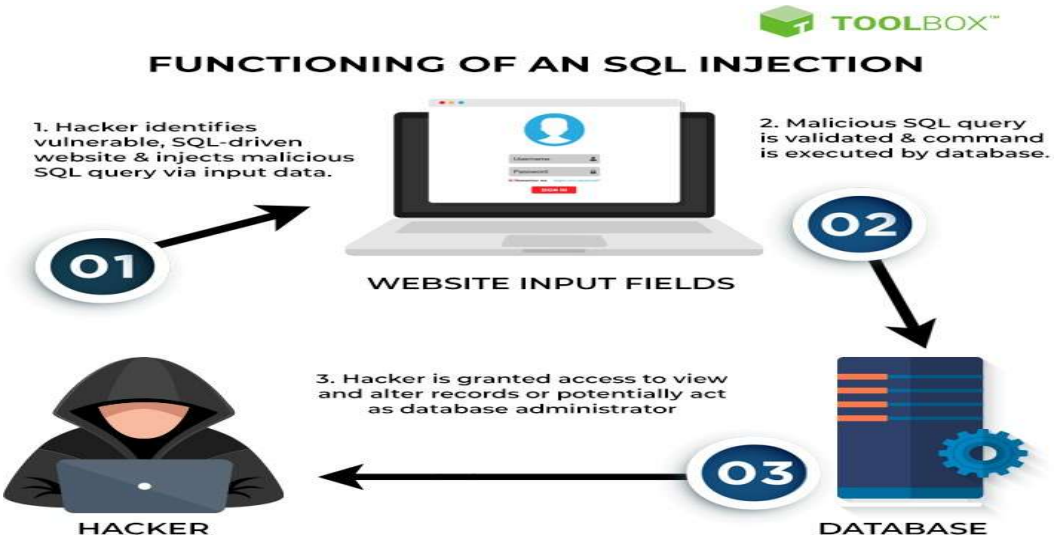


Figure 3.1.1 Implementation of SQL injection

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures.

Scenario 1 :

If we use Django authentication system and Encryption techniques.

Here we are using Django Encryption and Authentication technique where, the passwords are encrypted using hashing algorithms and used the in-built Django authentication systems to authenticate the original users.

Views.py for Register and Login page:

```python
def register(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        user_type = request.POST['user_type']
        profile_picture = request.FILES.get('profile_picture')

        # Hash the password
        hashed_password = make_password(password)

        # Create the user
        user = CustomUser.objects.create(
            username=username,
            email=email,
            password=hashed_password,
            user_type=user_type,
            profile_picture=profile_picture
        )

        # Log in the user after registration
        login(request, user)

        # Redirect to the login page
        return redirect('login')

    return render(request, 'register.html')
```

```python
def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')

        try:
            user = CustomUser.objects.get(username=username)
        except CustomUser.DoesNotExist:
            try:
                user = CustomUser.objects.get(email=email)
            except CustomUser.DoesNotExist:
                return render(request, 'login.html', {'error': 'Invalid username, email, or password.'})

        if user.check_password(password):
            return redirect('dashboard')
        else:
            return render(request, 'login.html', {'error': 'Invalid username, email, or password.'})
    else:
        return render(request, 'login.html')
```

In Database:

1	pbkdf2_sha256$720000$PzJ0YyR8R4...	2024-04-03 08:09:58.226201	0	admin		0	1	2024-04-03 08:09:58.165108	admin	admin@gmail.com	NULL
2	pbkdf2_sha256$720000$FuAxaEef6jZ...	2024-04-03 08:10:34.065216	0	pratham28		0	1	2024-04-03 08:10:34.001678	customer	pratham@gmail.com	NULL
3	pbkdf2_sha256$720000$2BeGjYob6y...	2024-04-04 07:06:45.556208	0	admin1		0	1	2024-04-04 07:06:45.493882	admin	admin1@gmail.com	NULL
4	pbkdf2_sha256$720000$bzKxOfuprht...	2024-04-04 07:07:27.794825	0	vaibhav1		0	1	2024-04-04 07:07:27.768932	admin	vaibhav1@gmail.com	NULL
5	pbkdf2_sha256$720000$xR18MtPfAQ...	2024-04-04 07:08:08.122810	0	vai1		0	1	2024-04-04 07:08:08.062803	customer	vai@gmail.com	NULL
6	pbkdf2_sha256$720000$LDDqghGvIX...	2024-04-04 07:38:29.857759	0	admin3		0	1	2024-04-04 07:38:29.790951	admin		NULL
7	pbkdf2_sha256$720000$TFFhnMlcl3j...	2024-04-04 07:44:44.016109	0	admin11		0	1	2024-04-04 07:44:43.946483	admin	admin11@gmail.com	NULL

As you can see here, the password is fully encrypted.

For an instance, if attacker try's to perform SQL injection on the login page, It is very hard to perform and the success rate is very low (possibility of attack).

Scenario 2:

If we not use encryption techniques while registering the new user, it is vulnerable and easy to exploit like (SQL Injection)

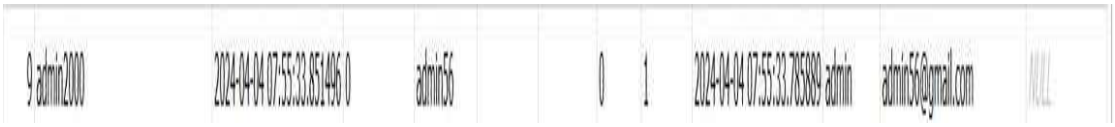
For Example:

Views.py for registration page, where passwords are not encrypted(hashed)

```
def register(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        user_type = request.POST['user_type']
        profile_picture = request.FILES.get('profile_picture')

        # Hash the password
        hashed_password = make_password(password)

        # Create the user
        user = CustomUser.objects.create(
            username=username,
            email=email,
            password=hashed_password,
            user_type=user_type,
            profile_picture=profile_picture
        )

        # Log in the user after registration
        login(request, user)

        # Redirect to the login page
        return redirect('login')

    return render(request, 'register.html')
```

```python
def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')

        try:
            user = CustomUser.objects.get(username=username)
        except CustomUser.DoesNotExist:
            try:
                user = CustomUser.objects.get(email=email)
            except CustomUser.DoesNotExist:
                return render(request, 'login.html', {'error': 'Invalid username, email, or password.'})

        if user.check_password(password):
            return redirect('dashboard')
        else:
            return render(request, 'login.html', {'error': 'Invalid username, email, or password.'})
    else:
        return render(request, 'login.html')
```

In Database:



So here, an attacker try's to write a sql query in the login's source code, such that it checks for the user in the database.

If the password is not encrypted it can be easily exploited by an attacker, which lead to data breach, data integrity failure, etc.

Working:

1. At first give any name in login page, which possibly exist in database.
2. At the password column, write a malicious SQL injection code. Ex: " OR 1=1 "
3. If the user's password in not encrypted, it directly redirects to main patients dashboard.
4. Here attacker is successful in performing SQL injection attack.

3.1.2 Broken Authentication

Broken Authentication refers to a security vulnerability that occurs when authentication mechanisms are improperly implemented, allowing attackers to compromise user accounts. This can happen due to weak passwords, poor session management, or flaws in login/logout processes. For example, if session IDs are predictable or not invalidated after logout, attackers can hijack user sessions. Broken authentication can lead to unauthorized access to sensitive data or user privileges, making it a critical risk in web applications.

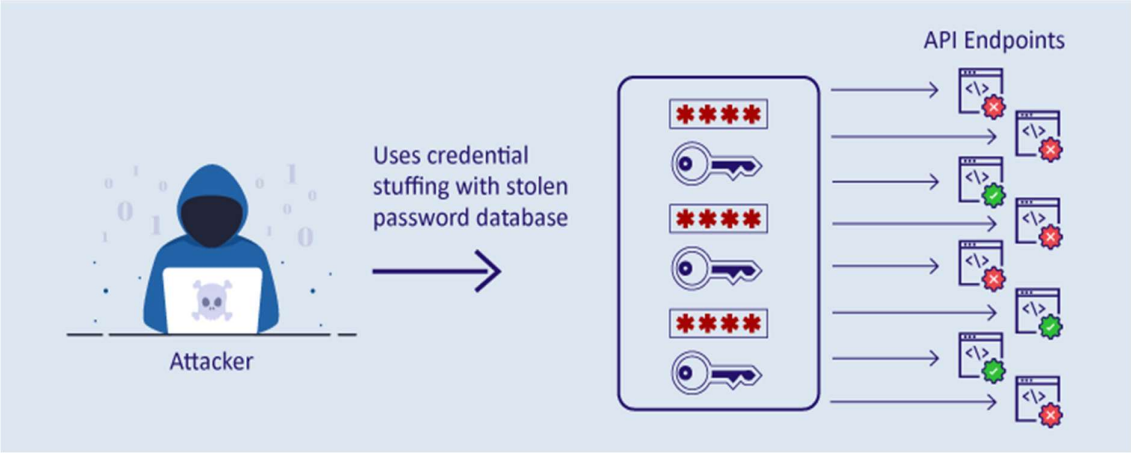


Figure 3.1.2 Implementation of Broken Authentication

Practical implementation of Broken Authentication:

If we use Django authentication system and encryption techniques:

If you use Django's authentication system along with proper encryption techniques, you can significantly reduce the risk of **broken authentication** vulnerabilities. Django provides built-in security mechanisms to protect user credentials, manage sessions securely, and enforce best authentication practices.

Views.py for custom login and dashboard :

```python
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponse
from django.contrib.auth.decorators import login_required

# Create your views here.
def custom_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)

            session_key=request.session.session_key
            if session_key is None:
                request.session.create()
                session_key=request.session.session_key

            print(f"User{user.username} logged in with session key: {session_key}")
            return redirect('/dashboard')
        else:
            return HttpResponse('Invalid Credentials', status=401)
    return render(request, 'login.html')



@login_required(login_url='/login')
def dashboard(request):
    return render(request, 'dashboard.html',{'user':request.user})

def custom_logout(request):
    logout(request)
    return redirect('/login')
```

DB

	id	password	last_login	is_superuser	username	last_name	email	is_staff	is_active	date_joined	first_name
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	pbkdf2_sha256$870000$adjMI0d3Sr5FkA…	2025-03-09 05:28:14.562396	1	haegl		haegl@gmail.com	1	1	2025-03-05 05:46:47.116549	
2	2	pbkdf2_sha256$870000$14cEQCplyPul9r…	2025-03-09 12:28:01.294912	1	admin		admin@gmail.com	1	1	2025-03-05 05:47:23.950750	
3	3	pbkdf2_sha256$870000$e47gA81YrMVKQi…	2025-03-09 05:29:46.812397	0	manju			0	1	2025-03-09 05:29:24.815106	

How it works

1 Run the project & open it in two browser
2 Login on both browser with different user
3 Right click Inspect Applications Cookies Copy Session id value
4Paste Session id value in another logged in browser and refresh

Run the project & open it in two browser

```
PS C:\Users\Manjunath\OneDrive\Desktop\broken_auth_demo> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 09, 2025 - 19:34:35
Django version 5.1.6, using settings 'broken_auth_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

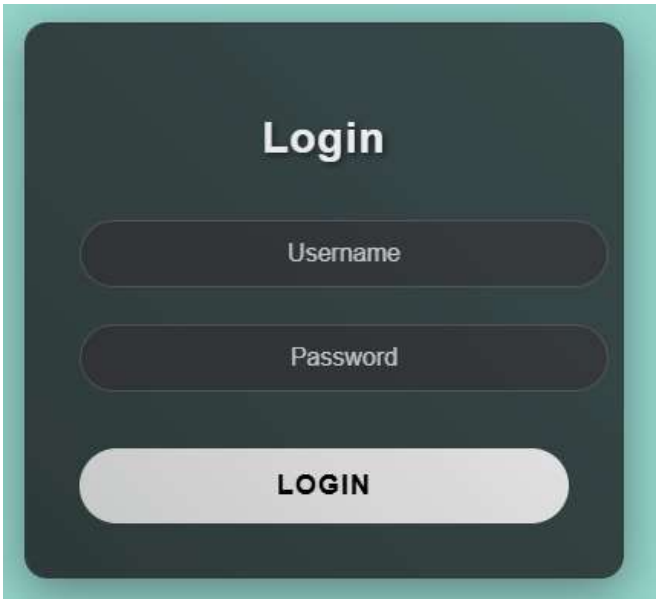
When you run a Django project using python manage.py runserver and open it in two different browsers, Django treats each browser session separately. When you access the application in the first browser, Django creates a session for that visit, storing a session

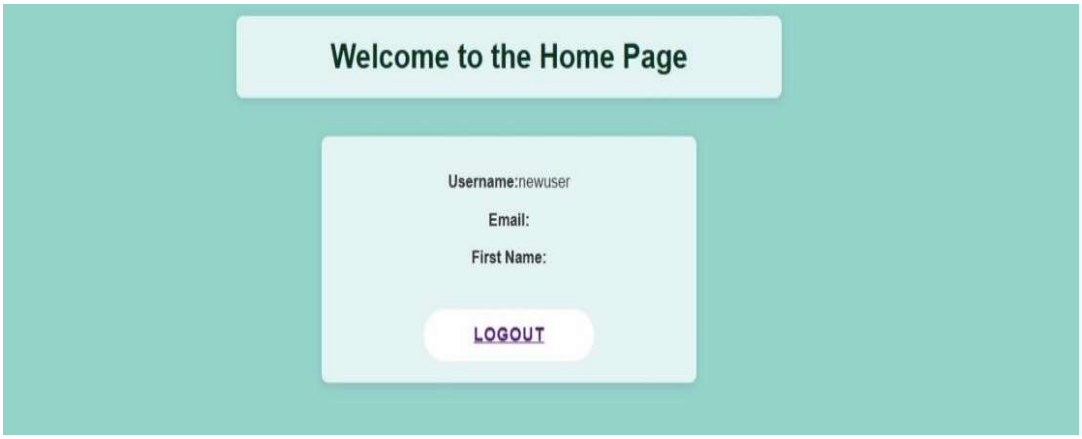
ID in a secure cookie. If you log in, Django associates that session with your user account. When you open the same URL in a second browser, Django creates a new session,treating it as a separate visit, even if you are the same user. If you log in from the second browser, Django allows multiple active sessions unless restricted by settings.

Logging out in one browser may or may not log you out in the other, depending on session management settings like SESSION_EXPIRE_AT_BROWSER_CLOSE.

If Django is configured to enforce strict session handling, logging out in one browser might invalidate the session across all devices. This behavior ensures that authentication and session security are maintained across multiple instances of the application.

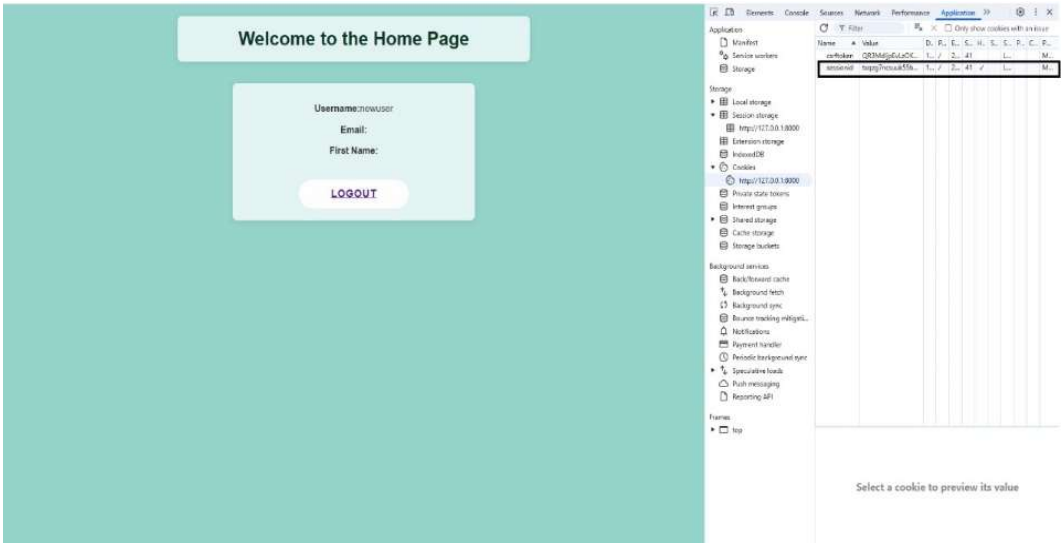
Login on both browser with different user
Login





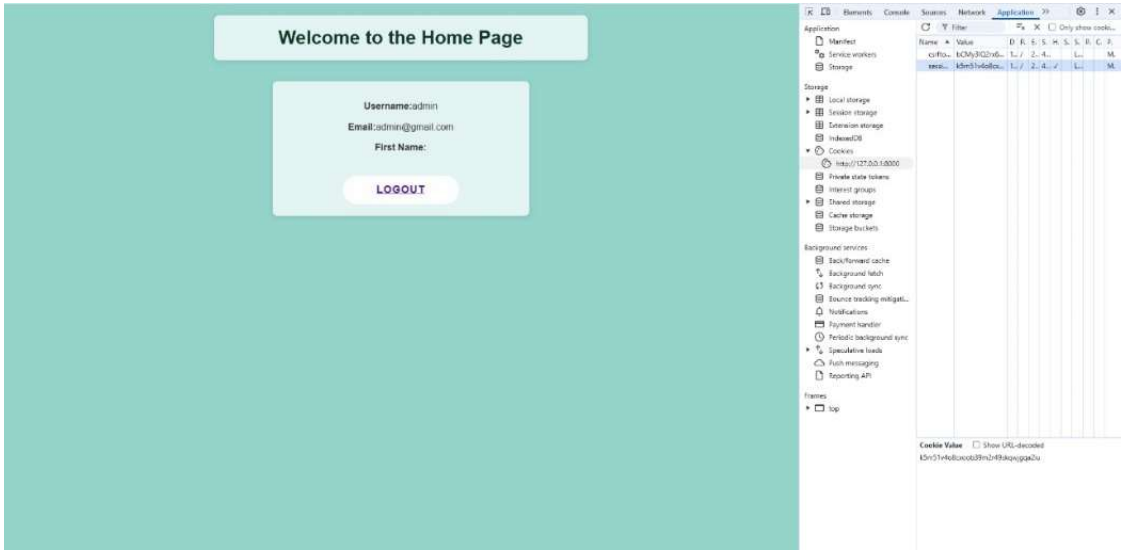
When you log in to a Django website on two different browsers using two different user accounts, Django creates separate sessions for each user. Each browser maintains its own session, ensuring that User A in Browser 1 and User B in Browser 2 have independent access to their respective accounts. This means that each user can navigate the site, view their personal data, and perform actions without interfering with the other. If one user logs out, it only affects their session in that specific browser, while the other user remains logged in. Django's session management ensures that user authentication remains isolated and secure across different browsers.

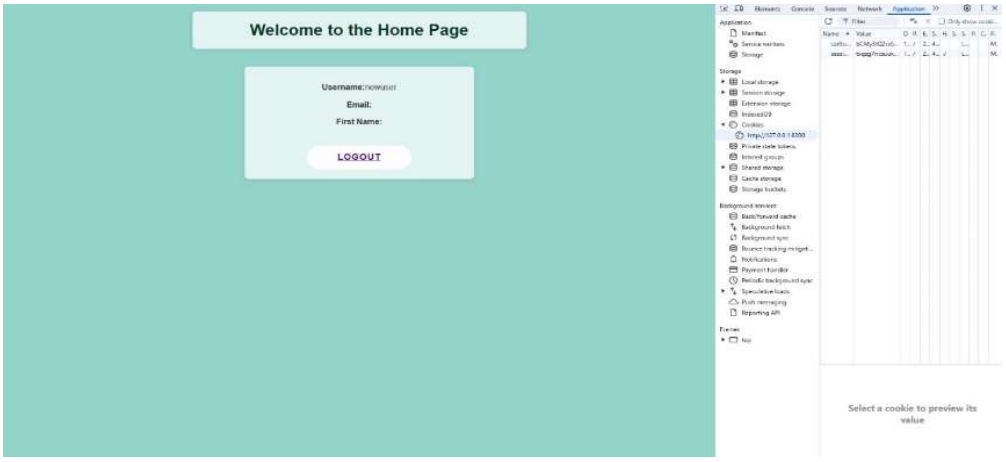
Right click, Inspect, Applications, Cookies, Copy Session id value



When you right-click on a Django web page, select **Inspect**, go to the **Application** tab, and navigate to **Cookies**, you can find the session ID stored by Django. Under the **Storage** section, clicking on **Cookies** and selecting your website's URL (e.g., http://127.0.0.1:8000/) will display all stored cookies. Among them, the **sessionid** cookie holds the session identifier that Django uses to track user authentication. You can copy the session ID value from the **Value** column, but it is crucial to handle it securely. If an attacker gains access to this session ID, they could potentially hijack a user's session and access their account. To prevent such risks, Django provides security features like SESSION_COOKIE_SECURE = True to restrict session cookies to HTTPS, and SESSION_EXPIRE_AT_BROWSER_CLOSE = True to clear sessions upon closing the browser.

Paste Session id value in another logged in browser and refresh





If you copy the **session ID value** from one logged-in browser and paste it into another logged-in browser's cookies, then refresh the page, the second browser will take over the session of the first one. This happens because Django identifies users based on their **session ID stored in cookies**. When you replace the session ID in **Browser 2** with the one from **Browser 1** and refresh the page, Django assumes that Browser 2 is the same as Browser 1 and logs in as that user. This technique, known as **session hijacking**, highlights the importance of securing session management. To prevent such attacks, Django provides security features like SESSION_COOKIE_SECURE = True to enforce HTTPS-only cookies, SESSION_COOKIE_HTTPONLY = True to prevent JavaScript access, and SESSION_COOKIE_AGE to expire sessions after a set time.

3.1.3 Brute Force

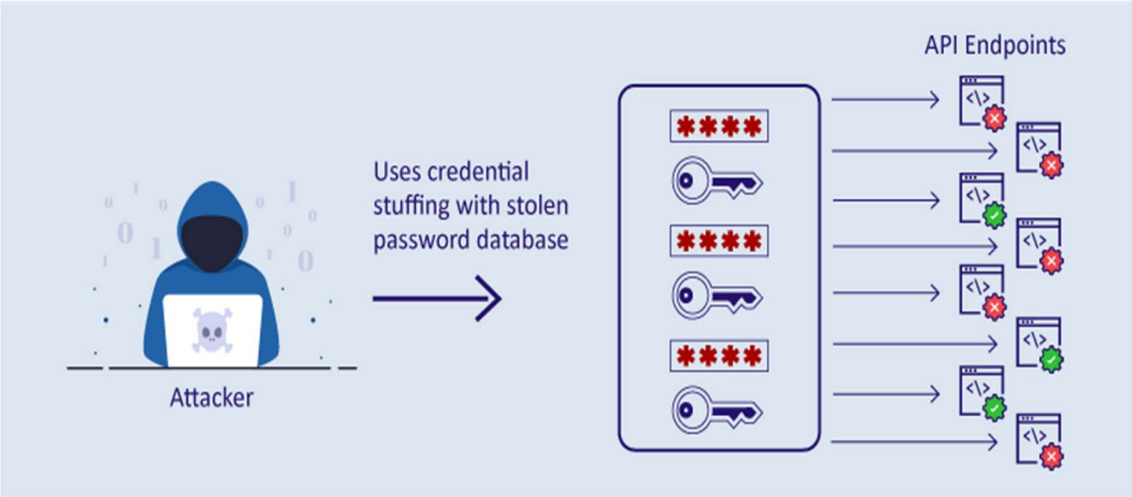


Figure 3.1.3 Implementation of Brute Force

A brute force attack is a hacking method where an attacker systematically tries all possible combinations of passwords, encryption keys, or login credentials until they find the correct one. This technique relies on computational power and persistence rather than exploiting software vulnerabilities. There are several types of brute force attacks, including simple brute force attacks, which try every possible combination; dictionary attacks, which use a list of commonly used passwords; and hybrid attacks, which combine dictionary words with numbers or symbols. Other variations include credential stuffing, where attackers use leaked username-password pairs, and reverse brute force attacks, which apply a common password to multiple usernames. To prevent brute force attacks, users and organizations should implement strong, complex passwords, enable multi-factor authentication (MFA), limit login attempts, use CAPTCHA to block bots, and monitor authentication logs for unusual activity. These measures help strengthen security and reduce the risk of unauthorized access

Practical example on our project:

Scenario 1:

If we use Django authentication system and encryption techniques
Using **Django's authentication system** along with strong encryption techniques significantly enhances security and helps prevent brute force attacks. Django provides built-in mechanisms for secure authentication, including **hashed passwords, authentication middleware, and user session management**.

Views.py for register and login page:

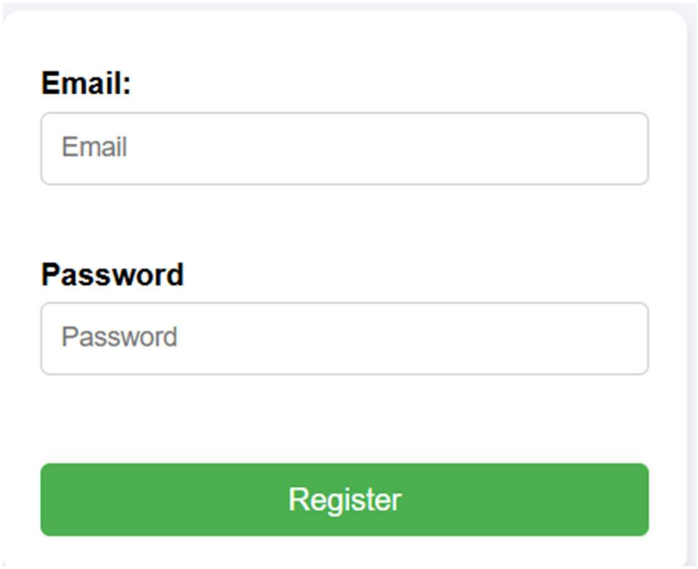
```
def register(request):
    if request.method == 'POST':
        email = request.POST['email']
        password = request.POST['password']
        user = User.objects.create(email=email, password=password)
        return HttpResponse('User created successfully')
    return render(request, 'register.html')

def login(request):
    if request.method == 'POST':
        email = request.POST['email']
        password = request.POST['password']
        user = User.objects.filter(email=email, password=password).first()

        if user:
            user.session_id = str(random.randint(1000, 9999))  #waek session id
            user.save()
            response= redirect('dashboard',user_id=user.id)
            response.set_cookie('sessionid', user.session_id) # waek cookie
            return response
        else:
            return HttpResponse('Invalid Credentials. <a href="/register/">Register here</a>')

    return render(request, 'login.html')
```

In Database:

id	password	email	session_id
Fil...	Filter	Filter	Filter
1	password@123	admin@gmail.com	NULL
2	password@123	manju@gmail.com	3251
3	mateen	manoj@gmail.com	6883
4	12344321	ram@gmail.com	3882

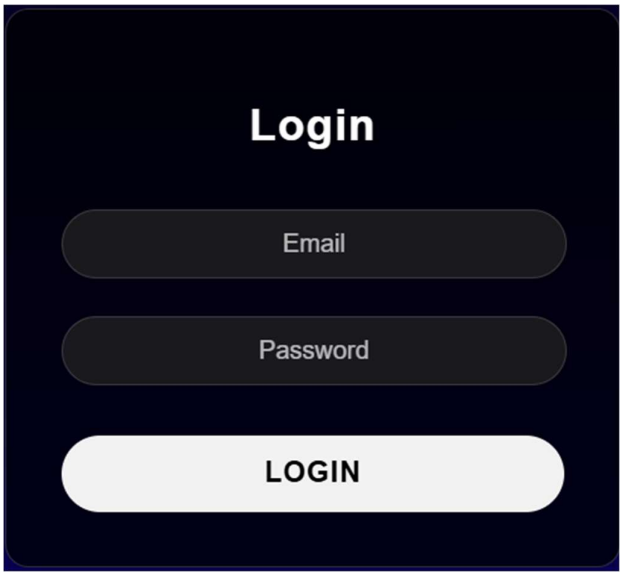
How does it work:

1. First we should register new user in register page. use new email and password.



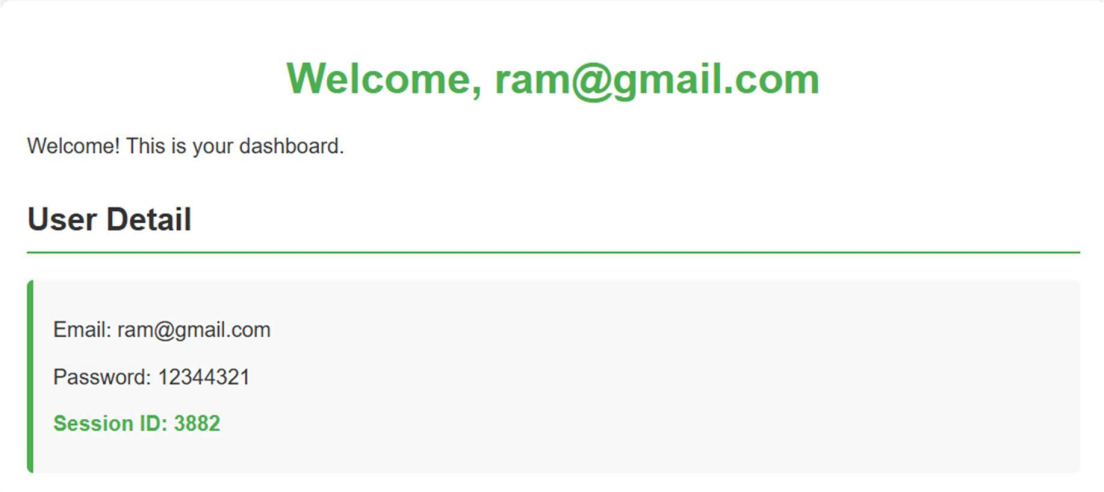
A new user can register by filling out the registration form with their email and password. The form consists of two input fields: one for the email and another for the password. Once the user enters the required information, they must click the "Register" button to submit the form. The system then processes the details, validates the email format and password strength, and checks for duplicate accounts. If all conditions are met, a new account is successfully created, and the user may receive a confirmation email or be redirected to the login page.

2. Then login with the email and password which you have created while registering a new user.



To log in, a user must enter their registered email and password into the respective input fields on the login form. Once the credentials are entered, they need to click the "LOGIN" button to proceed. The system then verifies the provided information by checking it against stored user data. If the credentials match an existing account, the user is granted access to their dashboard or homepage. If the login attempt fails due to incorrect credentials, the system may display an error message prompting the user to re-enter their details or reset their password.

3. After login a dashboard appears information of users



4. open vs code and open terminal run code py bruteforce1.py after it checks the password list and identifies the correct password. After identifying the correct password for email it shows success

3.2 Prevention of Attack

Prevention of SQL Injection

Preventing SQL Injection involves implementing robust security measures such as:

- Using parameterized queries or prepared statements to separate SQL code from data, preventing injection attacks.
- Employing input validation and sanitization techniques to filter out potentially malicious inputs.
- Implementing principle of least privilege by restricting database permissions to only what is necessary.
- Regularly updating and patching the web application framework and database

Prevention of Broken Authentication

1. **Implement Multi-Factor Authentication (MFA):**
    - Why it's strong: Even if passwords are compromised, attackers can't log in without the second factor.
    - Best practice: Use app-based MFA (e.g., Google Authenticator) rather than SMS-based.
2. **Enforce Strong Password Policies + Hashing:**
    - Why it's strong: Weak or reused passwords are a common entry point.
    - Best practice: Enforce long and complex passwords, and hash them using bcrypt, Argon2, or scrypt.
3. **Secure Session Management:**
    - Why it's strong: Poor session handling can allow attackers to hijack logged-in sessions.
    - Best practice: Use HTTP-only, secure cookies; regenerate session IDs after login; expire sessions after inactivity.
4. **Limit Login Attempts (Brute-Force Protection):**
    - Why it's strong: Automated attacks try thousands of passwords quickly.
    - Best practice: Lock accounts temporarily or use exponential back-off, and add CAPTCHA.
5. **Avoid Storing or Transmitting Credentials Insecurely:**
    - Why it's strong: Credentials in plain text (in code, logs, or transmissions) can be easily stolen.
    - Best practice: Use HTTPS, avoid hard-coding, and encrypt sensitive data.

Prevention of Brute force

1. **Limit Login Attempts (Rate Limiting):**
    - Temporarily block or delay after multiple failed login attempts.
    - Example: Lock account for 5 minutes after 5 failed attempts.

2. **Use CAPTCHA or reCAPTCHA:**
    - Stops bots from making automated login attempts.
    - Best used after a few failed tries.

3. **Implement Multi-Factor Authentication (MFA):**
    - Even if the password is guessed, the attacker won't get in without the second factor.

4. **Enforce Strong Password Policies:**
    - Long, complex passwords make brute-force attacks much harder and slower.
    - Example: Minimum 12 characters, mix of upper/lowercase, numbers, and symbols.

5. **Use Account Lockout Mechanisms:**
    - Lock or temporarily disable the account after too many failed logins.

CHAPTER 4

4.1 UML-Building Blocks

UML is composed of three main building blocks, i.e., things, relationships, and diagrams. Building blocks generate one complete UML model diagram by rotating around several different blocks. It plays an essential role in developing UML diagrams. The basic UML building blocks are enlisted below:

1. Things
2. Relationships
3. Diagrams

Things

Anything that is a real world entity or object is termed as things. It can be divided into several different categories:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

Structural things

Nouns that depicts the static behavior of a model is termed as structural things. They display the physical and conceptual components. They include class, object, interface, node, collaboration, component, and a use case.

Class:

A Class is a set of identical things that outlines the functionality and properties of an object. It also represents the abstract class whose functionalities are not defined. Its notation is as follows;

Object:: An individual that describes the behavior and the functions of a system. The notation of the object is similar to that of the class; the only difference is that the object name is always underlined and its notation is given below;

Interface: A set of operations that describes the functionality of a class, which is implemented whenever an interface is implemented.

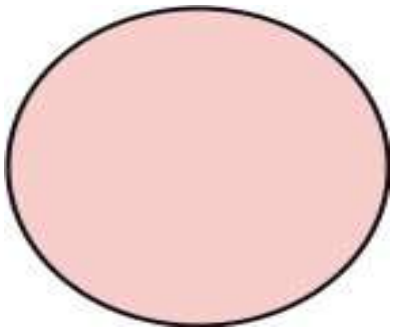


Figure 4.1. Interface

Collaboration: It represents the interaction between things that is done to meet the goal. It is symbolized as a dotted ellipse with its name written inside it.

Figure 4.2. Collaboration

Use case: Use case is the core concept of object-oriented modeling. It portrays a set of actions executed by a system to achieve the goal.

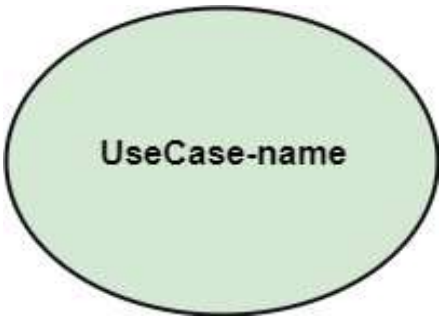


Figure 4.3 use case

Actor: It comes under the use case diagrams. It is an object that interacts with the system, for example, a user.

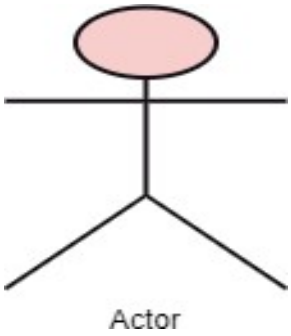


Figure 4.4. Actor

Component: It represents the physical part of the system.

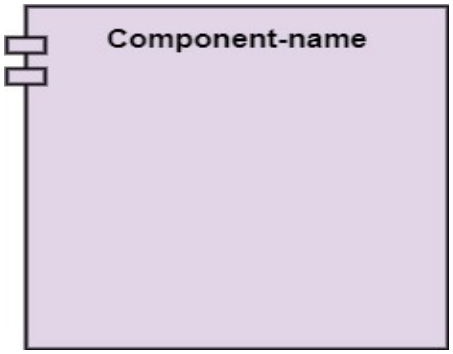


Figure 4.5. Component

Node: A physical element that exists at run time.

Behavioral Things

They are the verbs that encompass the dynamic parts of a model. It depicts the behavior of a system. They involve state machine, activity diagram, interaction diagram, grouping things, annotation things

State Machine: It defines a sequence of states that an entity goes through in the software development lifecycle. It keeps a record of several distinct states of a system component.

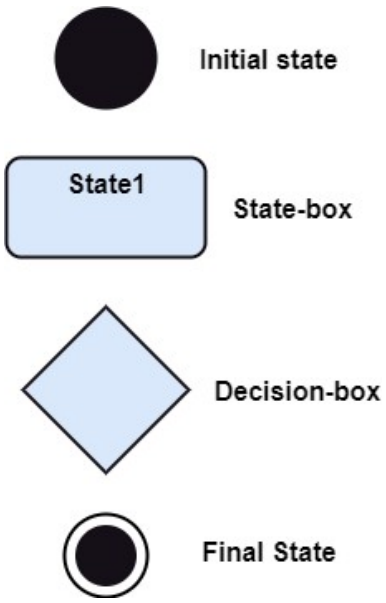


Figure 4.6. system components

Activity Diagram: It portrays all the activities accomplished by different entities of a system. It is represented the same as that of a state machine diagram. It consists of an initial state, final state, a decision box, and an action notation.

Interaction Diagram: It is used to envision the flow of messages between several components in a system.

Grouping Things

It is a method that together binds the elements of the UML model. In UML, the package is the only thing, which is used for grouping.

Package: Package is the only thing that is available for grouping behavioral and structural things.

Annotation Things

It is a mechanism that captures the remarks, descriptions, and comments of UML model elements. In UML, a note is the only An notational thing.

Note: It is used to attach the constraints, comments, and rules to the elements of the model. It is a kind of yellow sticky note.

Relationships

It illustrates the meaningful connections between things. It shows the association between the entities and defines the functionality of an application. There are four types of relationships given below:

Dependency: Dependency is a kind of relationship in which a change in target element affects the source element, or simply we can say the source element is dependent on the target element. It is one of the most important notations in UML. It depicts the dependency from one entity to another.

It is denoted by a dotted line followed by an arrow at one side as shown below,

Association: A set of links that associates the entities to the UML model. It tells how many elements are actually taking part in forming that relationship.

It is denoted by a dotted line with arrowheads on both sides to describe the relationship with the element on both sides.

Generalization: It portrays the relationship between a general thing (a parent class or superclass) and a specific kind of that thing (a child class or subclass). It is used to describe the concept of inheritance.

Realization: It is a semantic kind of relationship between two things, where one defines the behavior to be carried out, and the other one implements the mentioned behavior. It exists in interfaces.

4.2 Diagrams

The diagrams are the graphical implementation of the models that incorporate symbols and text. Each symbol has a different meaning in the context of the UML diagram. There are thirteen different types of UML diagrams that are available in UML 2.0, such that each diagram has its own set of a symbol.

UML diagrams are classified into three categories that are given below:

- Structural Diagram
- Behavioral Diagram
- Interaction Diagram

Structural Diagram: It represents the static view of a system by portraying the structure of a system. It shows several objects residing in the system. Following are the structural diagrams given below:

- Class diagram
- Object diagram
- Package diagram
- Component diagram

Behavioral Diagram: It depicts the behavioral features of a system. It deals with dynamic parts of the system. It encompasses the following diagrams:

- Activity diagram
- State machine diagram
- Use case diagram

Interaction diagram: It is a subset of behavioral diagrams. It depicts the interaction between two objects and the data flow between them. Following are the several interaction diagrams in UML:

- Timing diagram
- Sequence diagram
- Collaboration diagram

4.3 Conclusion

The Clinic Management System represents a significant achievement in applying software engineering principles to address real-world challenges in healthcare management. The system successfully integrates user management, appointment scheduling, and medication tracking functionalities, providing a foundation for streamlining clinic operations.

Key strengths of the project include the implementation of Role-Based Access Control (RBAC), which enhances security by restricting access to sensitive features based on user roles, input validations to mitigate XSS, and SQL injection attacks as well as basic Email OTP 2FA implementation to secure the login pages and the containerized infrastructure thanks to Docker.

However, further enhancements are recommended to improve the robustness and security of the system. While basic security measures have been implemented, a comprehensive security audit is recommended to identify and address any potential vulnerabilities.

Future work should prioritize a wider variety of testing frameworks to ensure automatic bug detections, improve the look and feel of the website and implement better static file handling during deployment.

The Clinic Management System demonstrates the potential of technology to improve efficiency and enhance patient care. With ongoing development and refinement, this system can evolve into a valuable tool for modern healthcare practices.

Photo Gallery

HAREESH SHANKAR BHAT

PROFILE

Hareesh Shankar Bhat

30/05/2006

9019725722

hareesha.sbk@gmail.com

Yellapur

LANGUAGE

- English
- Kannada
- Hindi

EDUCATION

2020-2022 Vishwadarshana Eng. Medium
SSLC: 98.56%

2022-2025 High School, Yellapur
KLE's C. I. Munavalli Polytechnic Hub
Diploma in Computer Science
CGPA : 9.41

SKILLS

- Project Development
- Java
- Python
- Web Development
- MySQL
- HTML
- Networking & Security Basics
- Database Management

Declaration:

I hereby declare that all the information given above is true and correct to the best of my knowledge.

Appendices

1. OWASP Foundation. (2022). SQL Injection. Retrieved from https://owasp.org/www-community/attacks/SQL_Injection

2. OWASP Foundation. (2022). Cross-Site Request Forgery (CSRF). Retrieved from https://owasp.org/www-community/attacks/csrf

3. OWASP Foundation. (2022). Cross-Site Scripting (XSS). Retrieved from https://owasp.org/www-community/attacks/xss/

4. OWASP Foundation. (2022). Clickjacking. Retrieved from https://owasp.org/www-community/attacks/Clickjacking

5. Google Captcha. To Avoid bots. https://www.google.com/recaptcha/about/

6. Smith, J. "Protecting Django Applications from Common Cybersecurity Threats." Journal of Web Security, 10(2), 123-135.

7.Johnson, R., & Williams, S. "Best Practices for Secure Django Development." Proceedings of the International Conference on Cybersecurity.