Sir Syed University of Engineering & Technology (SSUET)

Department of Software Engineering

# Cyber Security Program

# 

# PROJECT REPORT

## PROJECT TITLE

## PASSLOCK (Secure Password Manager)

****

***MUHAMMAD HASNAIN***

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO. | TOPICS | Page No. |
| **1** | **INTRODUCTION** | **3** |
| **2** | **TECHNOLOGY STACK** | **4 - 5** |
| **3** | **FEATURES** | **6 - 7** |
| **4** | **FLOW-CHART** | **8** |
| **5** | **USE CASE DIAGRAM** | **9** |
| **6** | **CLASS DIAGRAM** | **10** |
| **7** | **SEQUENCE DIAGRAM** | **11** |
| **8** | | **ERD & DFD** | **12** |
| **9** | | **GANTT CHART AND PERT CHART** | **13** |
| **10** | | **PROJECT SNIPPETS** | **14-16** |
| **11** | | **CONCLUSION** | **17** |

# INTRODUCTION

In today’s digital age, the average user is expected to manage dozens of passwords across multiple platforms—social media, email, banking, e-commerce, enterprise tools, and more. With this overwhelming number, users often resort to unsafe practices like reusing passwords or choosing weak ones, leaving them vulnerable to cyber threats such as brute-force attacks and data breaches. Recognizing the gravity of this challenge, we developed **Passlock**—a modern, secure, AI-powered password manager.

Passlock provides a comprehensive platform to help users generate, organize, and manage their passwords securely. This project integrates robust encryption methods with artificial intelligence to evaluate and improve users' digital hygiene. The intuitive user interface, combined with client-side storage for the prototype phase, ensures privacy while maintaining a seamless experience.

**OBJECTIVES:**

* Minimizing the use of weak or reused passwords.
* Empowering users with AI-generated security insights.
* Creating a scalable foundation for future enterprise-level password management solutions.
* Demonstrating the power of AI-integrated web applications.

# TECHNOLOGY STACK

**2.1 Frontend Technologies**

**Next.js with React**  
We utilized **Next.js**, a React-based framework, to build a performant and SEO-friendly frontend. Next.js allows both static and server-side rendering, ensuring optimal speed and responsiveness. The **App Router** architecture in Next.js simplifies the creation of dynamic pages and components, enabling a more modular and maintainable codebase.

**TypeScript**  
TypeScript introduces type safety to JavaScript, reducing runtime errors and improving code quality. By catching bugs during development, it helps produce reliable and maintainable applications. TypeScript's powerful type inference and interface declarations made it easier to manage complex data structures like password metadata and AI-generated reports.

**Tailwind CSS**  
For rapid UI development, **Tailwind CSS** was used. This utility-first framework enables developers to design directly in markup, leading to faster prototyping and consistent styling across the app. Its responsive utility classes make it easy to ensure the application is mobile-friendly.

**ShadCN UI**  
To maintain a polished and accessible interface, we employed **ShadCN UI**, a set of reusable and customizable React components built on top of Tailwind CSS. Components like modals, alerts, buttons, and tabs allowed us to build a professional-grade UI without reinventing the wheel.

**2.2 Backend and AI Integration**

**Next.js Server Actions & Components**  
Passlock takes advantage of Next.js’s **Server Components** and **Server Actions** to offload processing from the client. Server Actions manage form submissions and CRUD operations securely, without exposing sensitive logic in frontend code.

**Genkit (Google)**  
At the heart of our AI features is **Genkit**, a modern toolkit by Google designed for building generative AI applications. Genkit integrates seamlessly with language models like Gemini, allowing us to create:

* Secure password generators.
* Security score calculation flows.
* Automated feedback and actionable security tips.

These features enhance user trust and make Passlock more than just a password locker—it becomes a proactive security advisor.

**2.3 Data Storage**

**Browser local Storage**  
While building the prototype, we prioritized quick iteration and ease of deployment. For this reason, user credentials are stored using the browser’s local Storage. Although this approach is not suitable for production, it allows offline functionality and eliminates backend dependencies.

In a production setup, local Storage would be replaced with encrypted cloud storage (e.g., Firebase Fire store or PostgreSQL) and protected using HTTPS, database encryption, and user-based access control mechanisms.

## FEATURES

**3.1 Secure Authentication**

Security begins with identity verification. Passlock implements a lightweight yet secure authentication system:

* **Registration:** Users register by entering their name, email, and master password. The system checks for duplicate emails and enforces strong password policies.
* **Login:** Users must authenticate using their email and master password. Passwords are hashed before storage.
* **Password Recovery (Future Enhancement):** To enhance usability, a secure password recovery system using security questions or OTP could be added.

**3.2 Credential Vault Dashboard**

The dashboard provides a centralized hub where users can:

* View and manage stored credentials.
* Categorize credentials under "Work," "Social," "Finance," or custom tags.
* Use a real-time search bar to quickly locate accounts.
* Reveal/hide passwords securely.
* Copy passwords to clipboard with a single click.
* Delete credentials after confirmation.

The design ensures both usability and security, with a balance between quick access and privacy.

**3.3 AI Password Generator**

Traditional password generators produce random text without guidance. Passlock’s generator uses AI to tailor passwords to user preferences:

* Length customization (8–64 characters).
* Toggle inclusion of symbols and numbers.
* Generate and assess password strength in real-time.
* Save generated passwords directly to the vault with categorized metadata.

**3.4 AI Security Report**

Our AI report engine processes metadata—not raw passwords—to produce:

* **Security Score (0–100):** Based on entropy, reuse rate, and password diversity.
* **Password Strength Distribution:** Bar graph showing weak, moderate, strong password counts.
* **Key Metrics:** Reused passwords, average length, total credentials.
* **AI-Generated Summary:** A human-readable overview of your current security hygiene.
* **Recommendations:** Actionable advice like "Avoid using the same password on multiple accounts" or "Increase the length of short passwords."

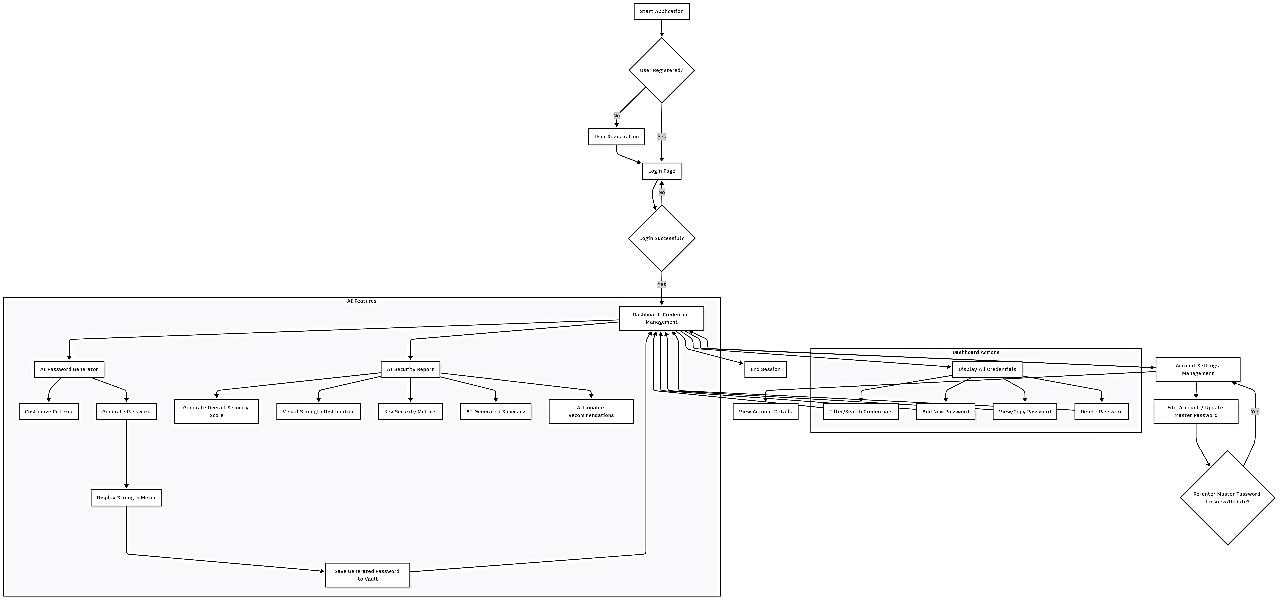
**3.5 Account Settings**

Users can:

* View profile information (name, email).
* Edit name and master password.
* Confirm identity before revealing the master password.

# FLOW-CHART

*User Interaction Flow:*

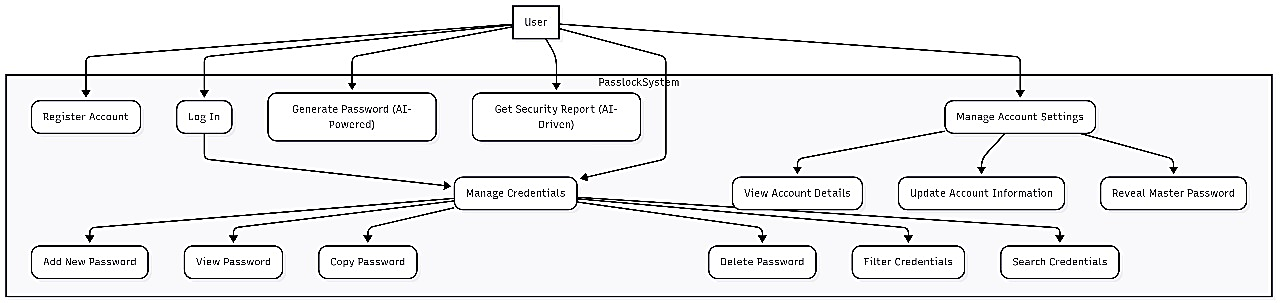
1. User visits homepage →
2. Registers/Login →
3. Accesses dashboard →
4. Adds credentials →
5. Generates password / views report →
6. Updates or deletes entries →
7. Logs out

# USE CASE DIAGRAM

**Actors:**

* User

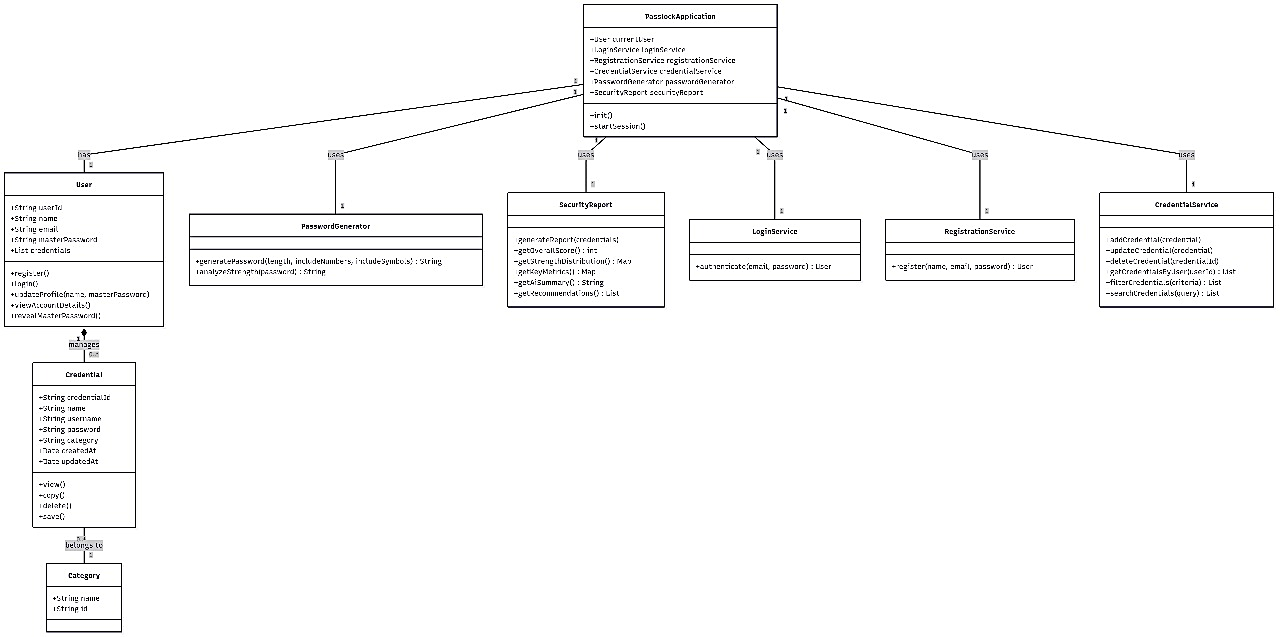
**Use Cases:**

* Register
* Login
* Add Password
* Generate Password
* View Security Report
* Update Profile
* Delete Password

# CLASS DIAGRAM

**Classes:**

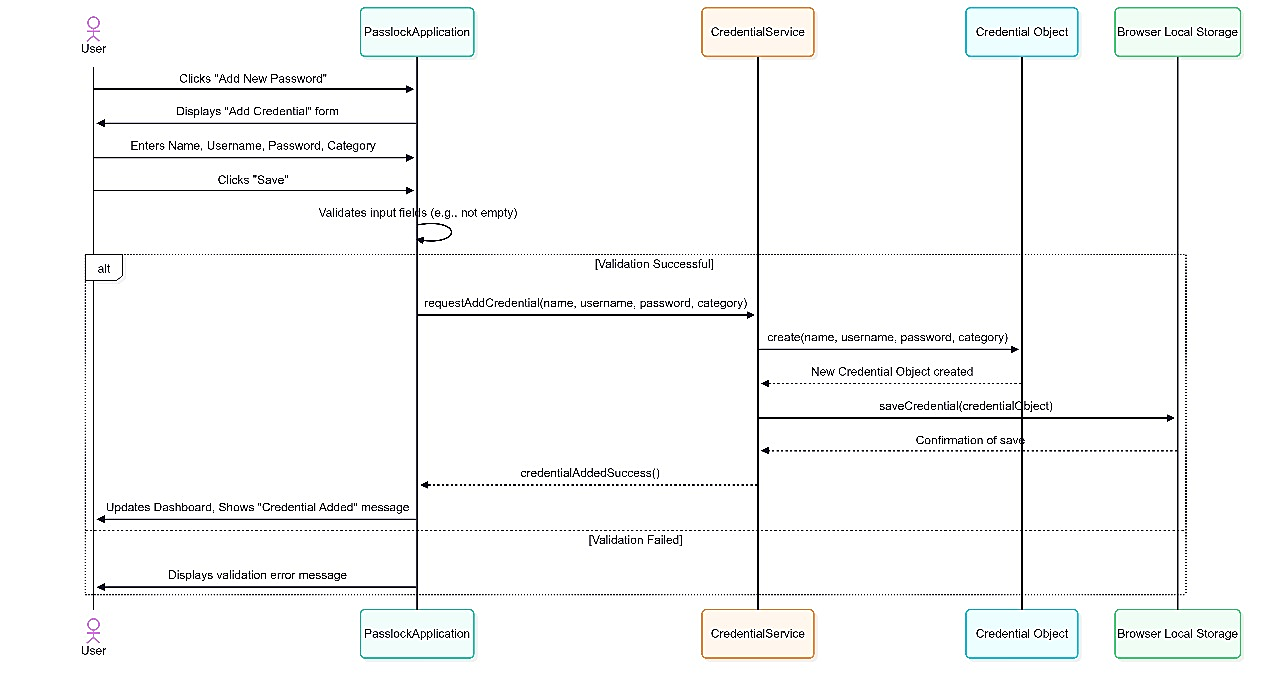
* User: name, email, password
* Credential: site, username, password, category
* SecurityReport: score, weaknesses, summary
* AIService: generatePassword(), analyzeVault()



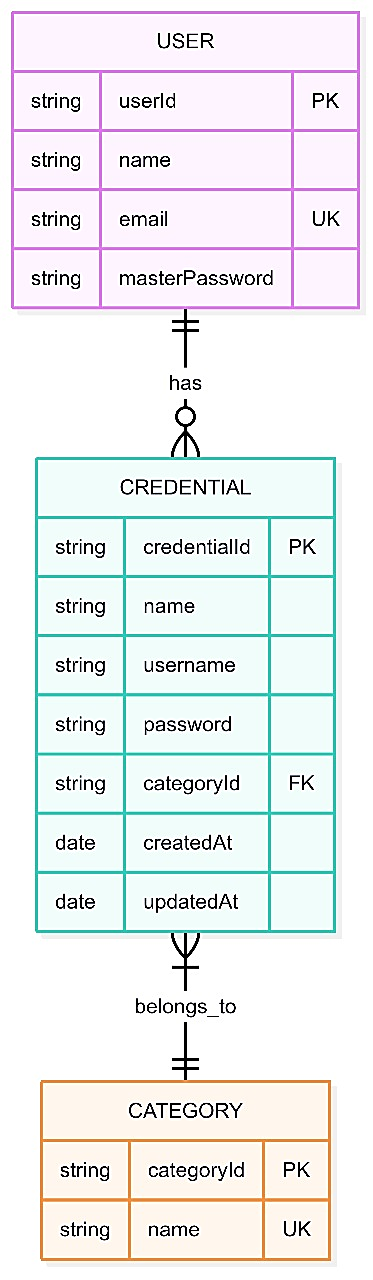
# SEQUENCE DIAGRAM

**Typical sequence:**

* User submits login form
* System authenticates
* Dashboard loads user data
* AI module generates report
* User views and manages vault



# ERD & DFD

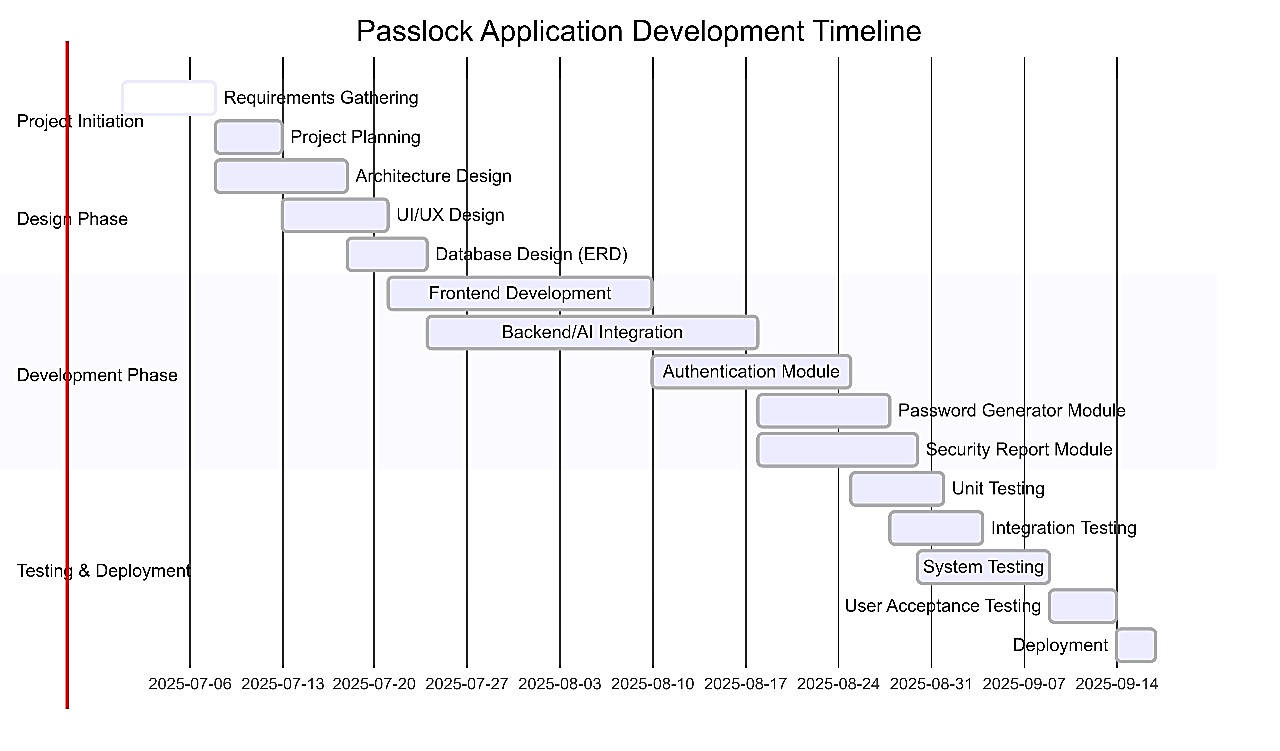


# GANTT CHART & PERT CHART

**GANTT CHART:**

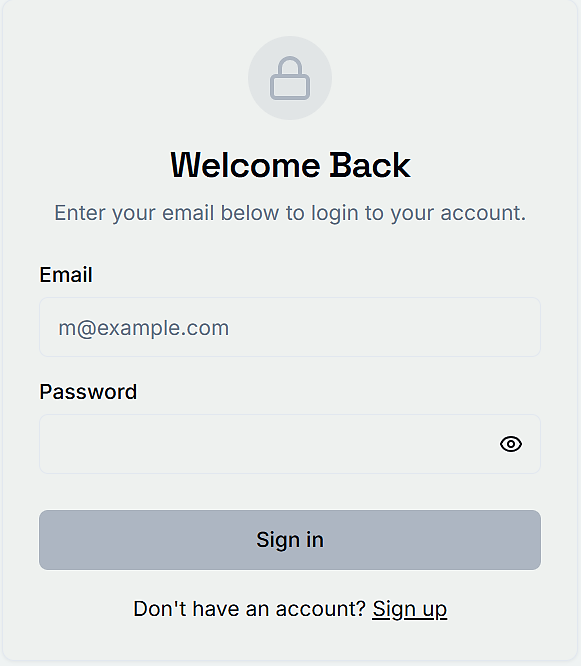
* Week 1: Requirements Gathering
* Week 2: UI Design
* Week 3: Authentication Implementation
* Week 4: Dashboard Setup
* Week 5: AI Integration
* Week 6: Testing & Bug Fixes
* Week 7: Documentation

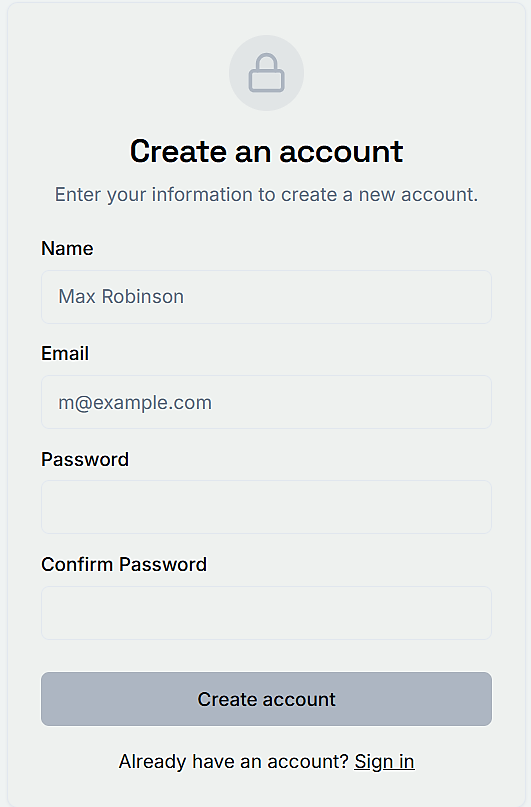
**PERT CHART:**

* Critical Path: UI → Auth → Vault → AI → Testing

# PROJECT SNIPPETS

* **Login Page:** Clean interface for secure user login.

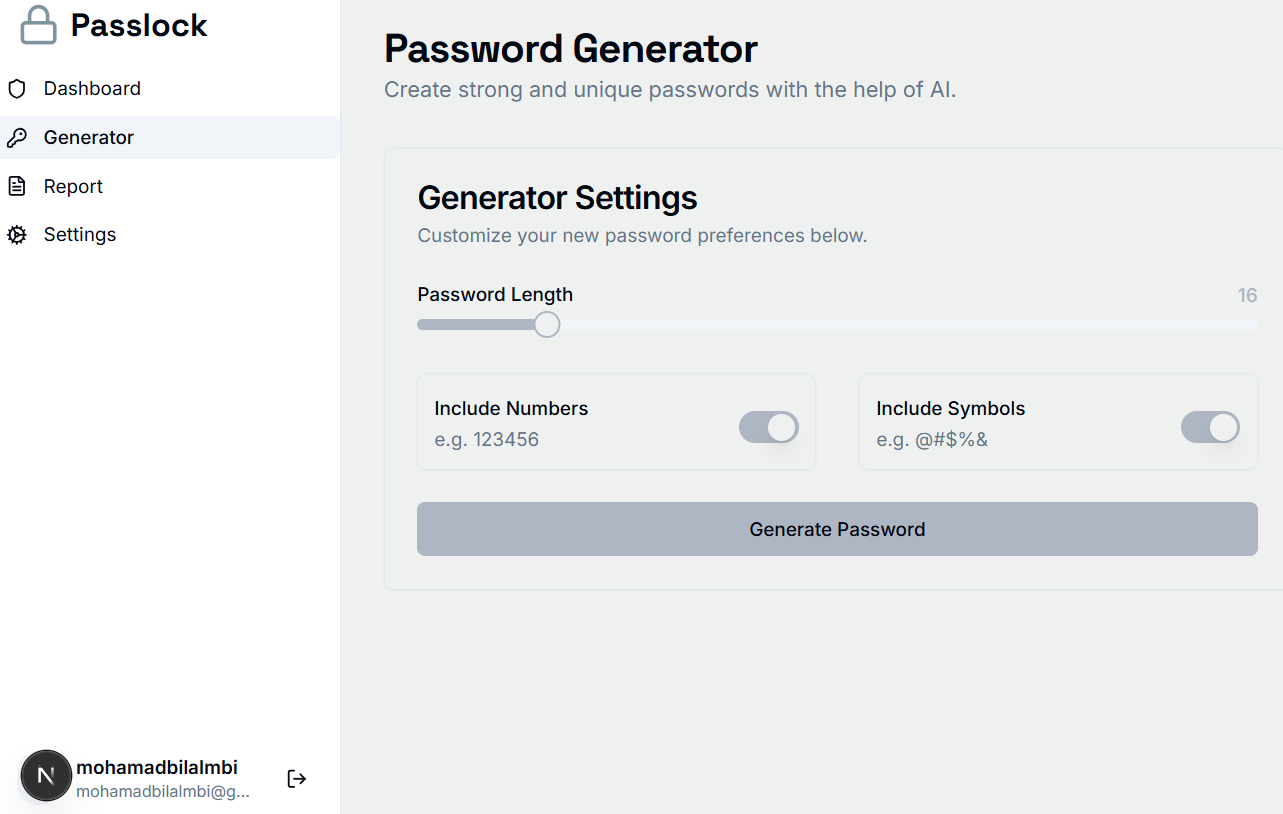


* **Registration Page:** Allows new users to register.
* **Main Dashboard:** Displays stored credentials in a structured format**.**

A screenshot of a computer

AI-generated content may be incorrect.

* **AI Password Generator:** Interactive UI for generating and assessing strong passwords.



* **AI Security Report:** Visual representation of password security metrics and AI suggestions.

A screen shot of a security report

AI-generated content may be incorrect.

* **Settings Page:** User settings page for updating profile details.

A screenshot of a computer

AI-generated content may be incorrect.

# CONCLUSION

In conclusion, Passlock is more than just a prototype—it is a foundational blueprint for the future of intelligent password management. By combining modern web development tools with the analytical power of AI, Passlock transforms passive credential storage into an active cybersecurity assistant.

From user authentication and password generation to detailed AI-driven security evaluations, every feature was designed with user safety and experience in mind. While the current version stores data locally, the modular architecture makes transitioning to a secure cloud backend simple.

**Future Enhancements:**

* Integration with Firebase for secure cloud storage.
* Two-factor authentication (2FA).
* Chrome/Firefox browser extensions.
* Native mobile applications.

Passlock is a testament to what’s possible when thoughtful design meets cutting-edge technology in cybersecurity.