Data Leak Prevention & Classification Project

Designing a system that uses AI for real-time data classification, leak detection and prevention to enhance data security.

Designed & Developed by:

Yeswanth Prasad, Owais Shariff, Prabhav Mishra and Pratham Vidhani

Project details

We were tasked with designing a project that tackles the following problem statement:

“System that uses AI for real-time data classification, leak detection and prevention to enhance data security.”

We decided that the way we would interpret and tackle this problem statement was to make it an end-to-end security solution for common people and/or small-scale organizations who are conscious enough about their digital security but don’t have the know-how to create a solution themselves or don’t want to spend money to buy a solution. Or even for just people who don’t trust third party providers with their personal data. The need for a comprehensive security solution tailored for everyday individuals and small-scale organizations who value digital security but lack the expertise to develop their own solution or prefer not to invest in third-party offerings due to privacy concerns seemed more apparent the closer we looked into it.

While our initiatives may not be groundbreaking or entirely proprietary, our vision centers on the recognition that even though excellent security solutions already exist, the majority of individuals lack the technical proficiency to utilize command-line tools designed for the same purpose. Furthermore, many people simply do not have the time to continuously monitor their personal security.

In response to these observations, our team aims to bridge the accessibility gap by providing user-friendly, all-in-one security solutions that cater to the unique needs of everyday users and small-scale organizations. We believe that the key to success lies in simplifying complex security measures and offering ongoing support and education to empower individuals and organizations to protect their digital assets. Our mission is to ensure that everyone, regardless of their technical background or available time, can enjoy the peace of mind that comes with robust digital security.

Our starting point was to identify the fundamental aspects that the average person must safeguard in order to enhance their digital security. This process led to the compilation of a comprehensive list, including:

- Implementing robust and secure passwords
- Avoiding password reuse
- Utilizing password-checking tools to identify potential data breaches.
- Assessing the vulnerability of one's IP address while browsing.
- Preventing IP and WebRTC leaks

With this in mind, we embarked on the development of Monolith: an end-to-end security solution crafted with the everyday user in mind.

Abstract

In response to the pressing need for robust digital security, we embarked on a project to create an end-to-end security solution, known as Monolith. This solution is tailored for common people and small-scale organizations who value their data security but lack the technical expertise or prefer to avoid third-party providers. Our mission centers on simplifying complex security measures, providing user-friendly tools, and offering continuous support and education to empower individuals and organizations.

Our project addresses fundamental aspects of digital security, including password management, prevention of password leaks, IP address vulnerability assessment, and protection against IP and WebRTC leaks. The development of Monolith is a direct outcome of our commitment to bridging the accessibility gap in the security realm, ensuring that anyone, regardless of their technical background or time constraints, can enjoy robust digital security.

Additionally, our project paper introduces a novel approach to password security by leveraging deep learning models. Traditional methods often fail to account for the possibility of password leaks. In contrast, our lightweight deep learning model, trained on a comprehensive password dictionary, can predict the likelihood of password leakage based on key features.

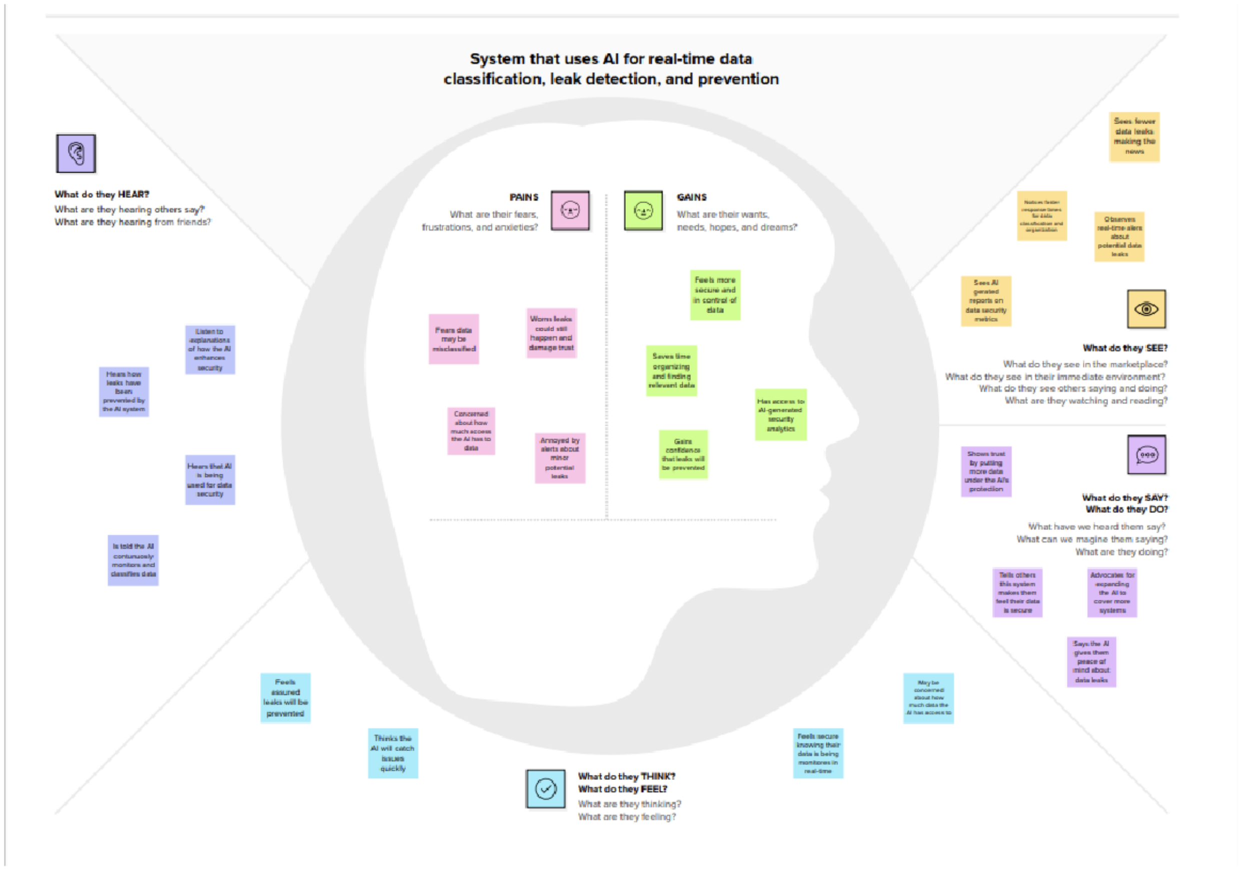
If you don’t prefer our dashboard, this model can be deployed on low-powered IoT devices, offering real-time password security assessment with an impressive 99% accuracy rate.

In summary, our project aims to make digital security accessible and efficient for everyone, and our paper presents an innovative solution for enhancing password security through deep learning technology.

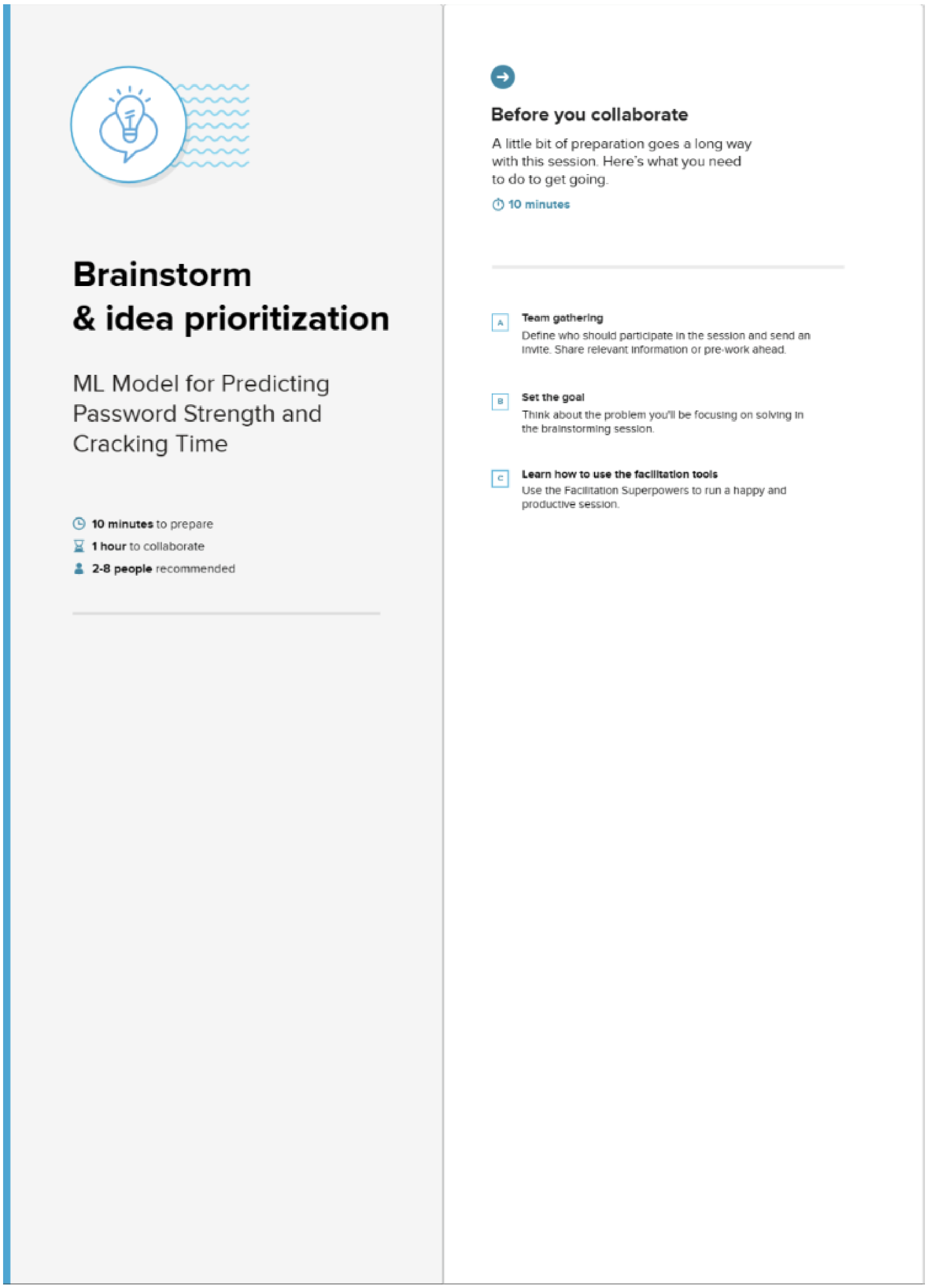
LITERATURE SURVEY

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Providing the user a complete interface for data leak prevention and digital identity protection and predicting the strength of passwords and estimating the time to crack them is important for evaluating and improving password security. However, doing this manually does not scale well.
2.	Idea / Solution description	We will create a machine learning model using TensorFlow and a sequential neural network architecture to automatically predict password strength as a classification problem and estimate cracking time via regression. Additionally we will use APIs that allow us to query passwords with K-Anonymity techniques to see if passwords have been pwned.
3.	Novelty / Uniqueness	Our solution is unique in its use of deep learning for this problem, which allows the model to learn complex password patterns. Fine-tuning the neural network architecture specifically for password data is a novel approach.
4.	Social Impact / Customer Satisfaction	More secure passwords mean better protection of user data and privacy And a dashboard will let the user have a easier interface to interact with. Our model can help guide users and systems to generate and adopt stronger passwords.
5.	Business Model (Revenue Model)	The model could be offered as a service to evaluate password security. Revenue could come from password auditing services or licensing the model itself.
6.	Scalability of the Solution	The TensorFlow implementation can scale to handle large datasets of password data. Once trained, the model can rapidly predict strength and cracking time for any input password. This is more scalable than manual or rules-based approaches. Additionally the database for storage is deployed in Docker which makes it easier to scale and the application itself is very portable

IDEATION & PROPOSED SOLUTION


System that uses AI for real-time data classification, leak detection, and prevention
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
Listen to explanations of how the AI enhances security
Hears how leaks have been prevented by the AI system
Hears that AI is being used for data security
Is told the AI continuously monitors and classifies data
PAINS
What are their fears, frustrations, and anxieties?
Fears data may be misclassified
Worry leaks could still happen and damage trust
Concerned about how much access the AI has to data
Annoyed by alerts about minor potential leaks
GAINS
What are their wants, needs, hopes, and dreams?
Feels more secure and in control of data
Saves time organizing and finding relevant data
Has access to AI-generated security analytics
Gains confidence that leaks will be prevented
Sees fewer data leaks making the news
Notices faster response times for data classification and organization
Observes real-time alerts about potential data leaks
Sees AI generated reports on data security metrics
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
Shows trust by putting more data under the AI's protection
What do they SAY?
What do they DO?
What have we heard them say?
What can we magine them saying?
What are they doing?
Tells others this system makes them feel their data is secure
Advocates for expanding the AI to cover more systems
Says the AI gives them peace of mind about data leaks
May be concerned about how much data the AI has access to
Feels secure knowing their data is being monitores in real-time
Feels assured leaks will be prevented
Thinks the AI will catch issues quickly
What do they THINK?
What do they FEEL?
What are they thinking?
What are they feeling?

Empathy maps like this have empowered our teams to connect with our target users, offering a structured framework for gathering and visualizing valuable insights about their thoughts, feelings, actions, and needs. These maps typically feature four quadrants that depict these crucial aspects, enabling us to dive into the users' perspectives. This, in turn, helps us craft products and solutions that are not just more relevant but also entirely centered around our users' needs and preferences.



Brainstorm & idea prioritization

ML Model for Predicting Password Strength and Cracking Time

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

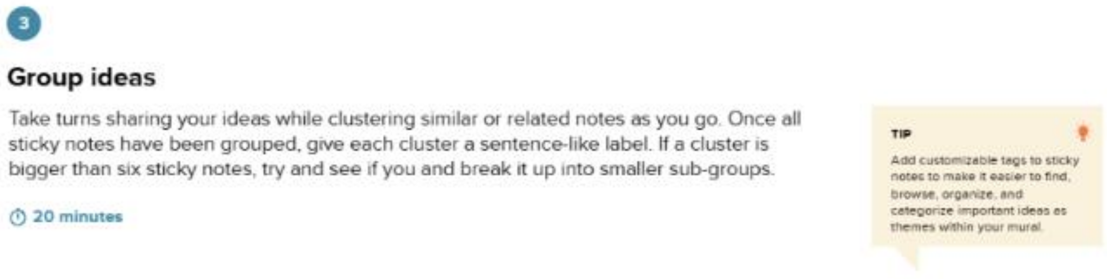
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Data Needed:

- Large dataset of passwords with associated labels for strength (e.g. weak, medium, strong)
- Data on time taken to crack passwords via brute force, associated with password
- Password features that may correlate with strength and cracking time, such as:
  - Length
  - Character types used (uppercase, lowercase, numbers, symbols)
  - Common passwords, keyboard patterns, dictionary words
- Other features like user info or context that may relate to password choices

Modeling:

- Try various ML algorithms - neural nets, random forests, SVMs
- For strength prediction, can frame as classification problem
- For cracking time prediction, frame as regression problem
- Evaluate different model performance, tune hyperparameters



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Challenges:

- Need large and representative dataset to train model effectively
- Strength and cracking time labels may be subjective or difficult to obtain
- Many features to evaluate for relevance to predictions
- Tradeoffs between model performance, interpretability, and efficiency

Next Steps:

- Find datasets of passwords with associated metadata
- Identify most relevant features to use
- Experiment with different ML models and hyperparameters
- Collect user feedback data to supplement training data
- Set up model validation and performance metrics
- Research explainability methods for model predictions
- Build functioning prototype and iterate

4

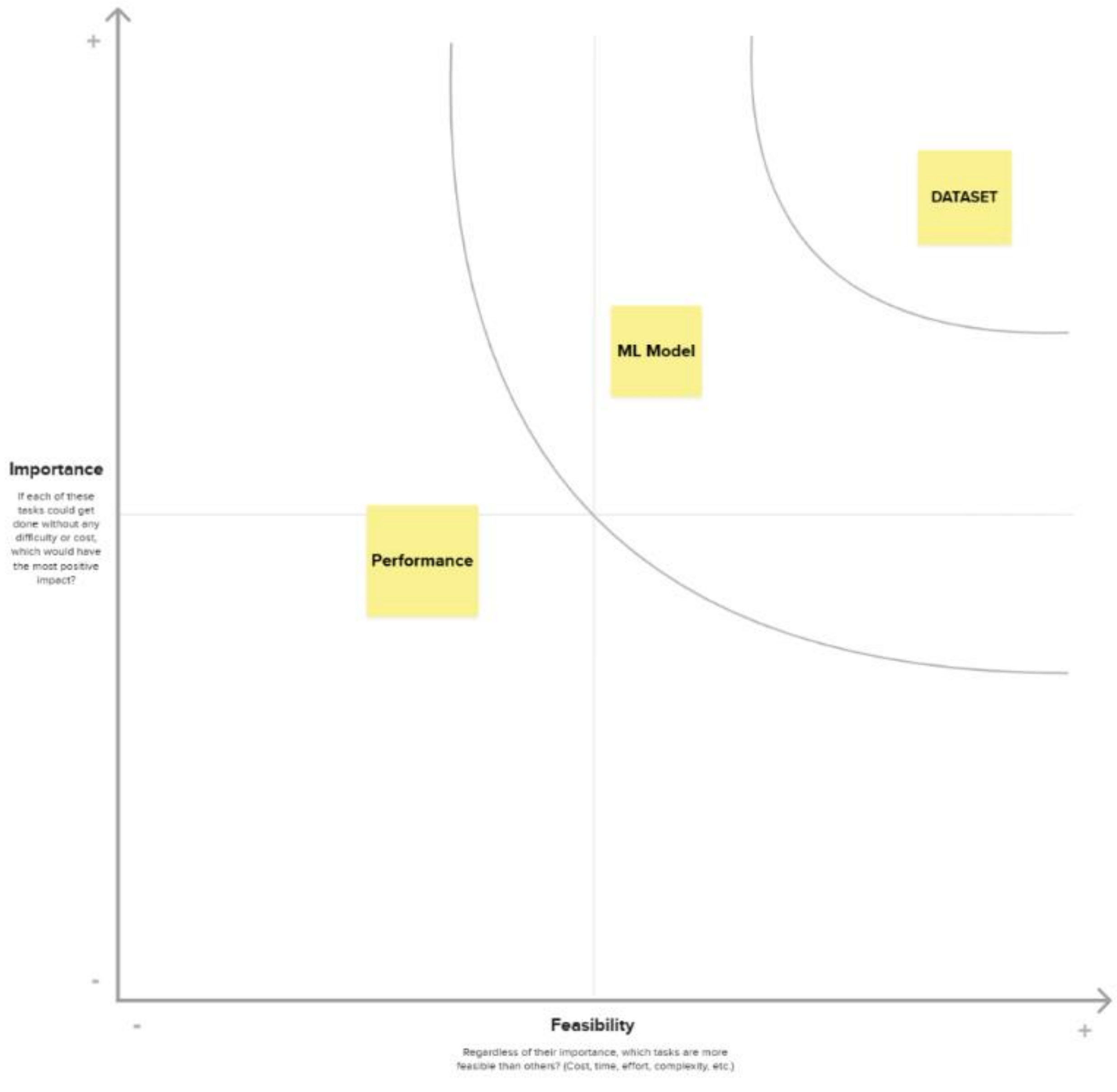
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

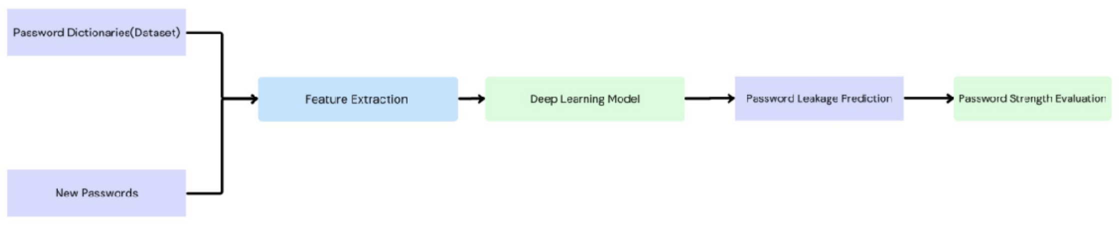
20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.


+
DATASET
ML Model
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Performance
-
-
Feasibility
+
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

PROJECT DESIGN


Password Dictionaries(Dataset)
New Passwords
Feature Extraction
Deep Learning Model
Password Leakage Prediction
Password Strength Evaluation

The diagram shows the steps involved in training and deploying the password leakage prediction model described in the paragraph. It starts with password dictionaries used to extract feature representations of the passwords. These password features are fed into a deep learning model that is trained to predict password leakage. The trained model can then be deployed to evaluate both the potential for leakage and overall security strength of new passwords based on their features. A key aspect is that the model runs locally on devices without needing external queries. This allows password evaluation to be performed efficiently even on low-powered IoT devices. The modular flow from passwords to feature extraction to model training and finally password scoring enables the development and deployment of an on-device password security assessment tool using deep learning

Data Collection and Preprocessing

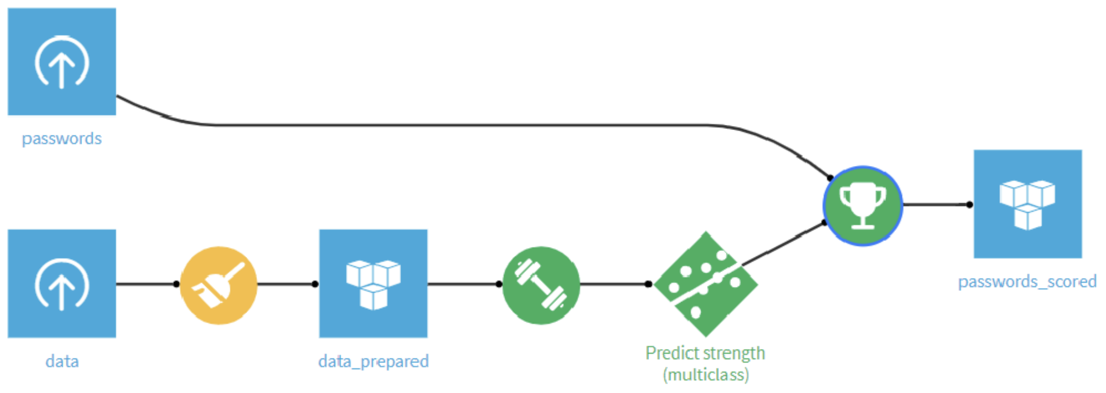
- Collect datasets of passwords and associated metadata like strength labels, cracking times, etc.
- Clean and preprocess data - handle missing values, normalize features, encode categorical variables, etc.
- Split data into training, validation, and test sets.
- Model Development and Training
- Design and configure the TensorFlow sequential model architecture - number of layers, nodes, activation
- functions, etc.
- Train model on preprocessed password data for both classification (strength) and regression (cracking time)
- tasks
- Validate model on holdout data and tune hyperparameters for optimal performance.

Model Deployment

- Package final optimized models and make available via API for integration into applications.
- Create web interface/dashboard for direct model access.
- Containerize model with Docker for scalable deployment on cloud or local systems.

Production Integration

- Integrate API into password checking systems to score and make recommendations.
- Integrate web interface into password manager applications or security admin consoles.
- Monitor and collect performance metrics, feedback data to improve model.
- The TensorFlow model at the core enables high accuracy and scalability. The web and API services allow flexible integration into production password systems to operationalize the predictions.



This diagram illustrates how In the model development and training phase, the TensorFlow sequential model architecture is carefully designed and configured. Parameters such as the number of layers, nodes, and activation functions are optimized for model performance. The model is trained using the preprocessed password data to accomplish both classification (for assessing password strength) and regression (for estimating cracking time) tasks. Model validation is performed on holdout data, and hyperparameters are fine-tuned to ensure optimal performance.

After successful model development and optimization, the final models are packaged and made available via API, facilitating seamless integration into various applications. A user-friendly web interface and dashboard are also created, allowing direct access to the model's capabilities. To ensure scalable deployment, the model is containerized with Docker, enabling deployment on cloud or local systems as needed.

The production integration phase involves incorporating the API into password checking systems, enabling real-time scoring and recommendations. The web interface is integrated into password manager applications and security admin consoles, enhancing user accessibility. Ongoing monitoring and collection of performance metrics help in refining the model continuously, with user feedback data used to further improve model accuracy and effectiveness.

REQUIREMENT ANALYSIS

Challenges Faced in Development

The development of a solution architecture for a password strength and cracking time prediction system involves various complex stages, each with its unique set of challenges. In this section, we will delve into the challenges faced during the implementation of the described architecture.

Data Collection and Preprocessing Challenges

1. **Data Quality**: One of the primary challenges in this phase is ensuring data quality. Password datasets can be noisy, and issues like incorrect labels, missing values, or outliers can significantly impact the model's performance. Cleaning and preprocessing data to ensure accuracy and consistency is crucial but time-consuming.

2. **Data Imbalance**: Password strength labels may not be evenly distributed across the dataset. This imbalance can lead to biased models, as the algorithm may favor the majority class. Techniques such as oversampling, under sampling, or the use of different loss functions are required to address this issue.

3. **Feature Engineering**: Encoding categorical variables and normalizing features require careful consideration. Choosing the right encoding schemes and normalization techniques can be challenging, as they can affect model performance. It's important to strike a balance between maintaining information and avoiding overcomplicated feature transformations.

Model Development and Training Challenges

1. **Overfitting**: Overfitting is a significant concern during model development. Deep learning models, such as those using TensorFlow, can be highly flexible, which increases the risk of overfitting. Proper regularization techniques, like dropout and L2 regularization, need to be employed to mitigate this issue.

2. **Hyperparameter Tuning**: Configuring the architecture of the neural network, including the number of layers, nodes, and activation functions, is not a straightforward task. Finding the optimal hyperparameters can be challenging, and grid search or random search techniques are often necessary to explore a wide range of possibilities.

3. **Computational Resources**: Training deep learning models can be computationally intensive, especially if working with large datasets and complex architectures. Ensuring access to adequate computational resources, such as GPUs or TPUs, can be a logistical challenge, particularly for smaller organizations.

4. **Interpretability**: Deep learning models are often considered "black boxes" because of their complexity. Interpreting the decisions made by these models can be challenging, which is a concern in fields where model interpretability is critical, such as healthcare or finance. Developing methods to make deep learning models more interpretable is an ongoing research area and a deployment challenge.

5. **Model Size and Deployment**: Deploying deep learning models on edge devices or in resource-constrained environments can be a significant challenge. Shrinking the model size while maintaining performance is a common requirement for such deployments. This involves techniques like model quantization and compression to reduce memory and processing demands.

Model Deployment Challenges

1. **API Development**: Creating an API for model integration is a non-trivial task. It involves considerations such as API security, handling requests, and ensuring uptime. Scaling the API to handle a large number of requests and maintaining it can be challenging, especially when dealing with concurrent user access.

2. **Web Interface/Dashboard**: Developing a user-friendly web interface or dashboard is a design and development challenge. It involves UI/UX considerations, responsiveness, and the integration of the model's predictions. Ensuring a seamless user experience across different devices and browsers is crucial.

3. **Containerization**: While containerizing models with Docker offers scalability, it also comes with challenges, such as ensuring compatibility across different environments and dealing with potential containerization issues.

4. **Data Privacy and Security**: Deploying machine learning models often involves handling sensitive data. Ensuring data privacy and security is a significant challenge. Implementing encryption, access controls, and compliance with data protection regulations are crucial aspects of model deployment.

5. **Monitoring and Maintenance**: Once a model is deployed, it requires continuous monitoring and maintenance. Detecting and addressing performance degradation, drift in data distributions, and other issues are essential to ensure the model's reliability and accuracy over time. This involves setting up alerting systems, automated checks, and scheduled maintenance tasks to keep the model functioning at its best. Regular updates to the model and its dependencies are also necessary to address issues and improve performance as new data becomes available.

Product Integration Challenges

1. **Integration into Existing Systems**: Integrating the API into password checking systems and web interfaces into password manager applications or security admin consoles can be complex. It requires a deep understanding of the target systems, including their architecture, security requirements, and compliance standards.

2. **Monitoring and Feedback**: Collecting and analyzing performance metrics, as well as user feedback, is crucial for model improvement. Setting up an efficient monitoring system to capture real-time data and derive meaningful insights from it is challenging and requires continuous effort.

3. **Scalability**: As the demand for the solution grows, ensuring its scalability can be challenging. This involves load balancing, resource allocation, and redundancy planning to maintain high availability and performance.

4. **Data Privacy and Security**: When integrating an AI model into a Next.js-based dashboard for password management or security administration, data privacy and security become paramount. Ensuring that sensitive user information and authentication data are adequately protected is a significant challenge. This includes encryption, access control, and compliance with data protection regulations like GDPR or HIPAA, depending on the context.

5. **User Experience**: The user experience is crucial when implementing AI in a dashboard. Balancing the AI's decision-making with user-friendly interfaces and ensuring that users can easily understand and interact with the AI-driven features can be a complex task. This may involve user testing and iterative design improvements.

6. **Customization and Adaptation**: Every organization has unique requirements for their password manager or security dashboard. Adapting the AI model to meet these specific needs can be challenging, as it may require customization, fine-tuning, and integration with other services or APIs.

7. **Training and Maintenance**: AI models require continuous training and maintenance to stay effective. Developing a robust strategy for retraining the model, handling updates, and addressing issues as they arise is an ongoing challenge. This includes keeping the AI model up-to-date with evolving security threats and password management best practices.

8. **Compatibility with Next.js**: Ensuring that the AI model seamlessly integrates with the Next.js framework can be a technical hurdle. Compatibility issues may arise in terms of code structure, data communication, and performance optimization, which need to be addressed to make the integration work smoothly.

9. **Regulatory Compliance**: Depending on the industry and region, there may be specific regulatory requirements that need to be met when integrating AI into security or password management systems. Navigating these compliance challenges, such as SOC 2, ISO 27001, or industry-specific standards, can be complex and time-consuming.

10. **Collaboration and Skillsets**: Integrating AI into a Next.js-based dashboard often requires collaboration between AI specialists, front-end and back-end developers, security experts, and compliance professionals. Ensuring effective communication and cooperation among these teams with different skillsets can be a challenge in itself.

PROJECT PLANNING & SCHEDULING

Features of the Final Solution (Marketable)

1. **User Authentication Using Firebase**

Our Password Manager takes user authentication to the next level by seamlessly integrating with Firebase. This partnership ensures a reliable and user-friendly authentication system, where account creation, login, and password resets are a breeze. Firebase's authentication services provide robust user identity management, incorporating multi-factor authentication options that enhance the security and trustworthiness of user accounts. With our Password Manager, your users' data is in safe hands, protected by one of the most trusted authentication services available.

2. **Docker Container for Database Deployment**

Simplify your database deployment with our innovative Docker container support. This feature streamlines the process of deploying your database, reducing the complexity and compatibility concerns that often plague database management. Our containerization approach guarantees flexibility, scalability, and consistency across different environments. This means that you can focus on managing your data effectively, without worrying about deployment-related hassles.

3. **AI Password Analysis (TensorFlow Model)**

Incorporating cutting-edge AI technology powered by TensorFlow, our Password Manager offers a revolutionary password analysis feature. This AI-driven component evaluates the strength and security of passwords in real-time, providing users with immediate feedback on their password choices. This proactive approach to password security empowers users to make informed decisions, strengthening their defenses against potential threats. With our Password Manager, security is not just a feature; it's a promise.

4. Secure Password Storage

Security is paramount in our Password Manager. We employ industry-standard encryption algorithms to ensure that all stored passwords are safeguarded from unauthorized access. Even system administrators cannot decipher these passwords, guaranteeing that your users' credentials remain confidential and secure. Our commitment to data security goes beyond the surface, offering peace of mind to both you and your users.

5. Password Generation

Creating strong and unique passwords has never been easier with our built-in password generation feature. Users can generate complex passwords with customizable options for length, character types, and special symbols. This functionality eliminates the need for users to come up with their own passwords, effectively reducing the risk of weak or easily guessable choices. With our Password Manager, strong password creation is just a click away.

6. Password Retrieval

Forgot your password? Don't worry; our Password Manager has got you covered. We provide a convenient password retrieval mechanism that allows users to request password reset links via email. This process ensures that users can quickly and securely regain access to their accounts, minimizing downtime and reducing user frustration.

7. Password Management

Simplify your digital life by managing all your passwords in one convenient location. Our Password Manager offers a user-friendly dashboard for organizing and categorizing passwords. Users can effortlessly update, delete, or add new passwords, streamlining the management of multiple accounts and credentials. With our Password Manager, you'll have complete control over your digital identity, making it easier than ever to keep track of your online security.

Technical Stack

When it comes to building a robust and secure Password Manager project, selecting the right tech stack is paramount. Each component plays a crucial role in ensuring the functionality, security, and user experience of the application. In this document, we will delve into our choice of technologies, explaining why we believe they are the best fit for our Password Manager project.

NextJS: Powering Seamless User Interfaces

Our choice of NextJS as the frontend framework stems from its ability to provide fast and responsive user interfaces, making it an ideal choice for a Password Manager project. Its server-side rendering capabilities enhance performance and search engine optimization, ensuring that our application is not only secure but also user-friendly. Additionally, the built-in TypeScript support ensures code reliability, making development a more efficient and safer process.

TypeScript: Enforcing Code Quality and Safety

TypeScript is a language choice that aligns perfectly with our commitment to code quality and security. By providing static typing, TypeScript helps us catch errors early in the development process, reducing the likelihood of security vulnerabilities and runtime issues. The type system promotes better code documentation and maintains code quality, which is crucial in a security-focused application like a Password Manager.

TailwindCSS: Streamlined Styling and Consistency

TailwindCSS is our preferred CSS framework for several reasons. Its utility-first approach allows us to rapidly create a cohesive and consistent user interface, a critical element in a password management application. This approach makes it easier to maintain a sleek and user-friendly design while ensuring that the application remains responsive and functional across various screen sizes and devices.

Firebase: A Secure and Scalable Backend

When it comes to the backend, Firebase is the backbone of our Password Manager project. Its integration with our frontend ensures secure user authentication, a vital feature for a Password Manager application. Firebase offers a range of authentication services, including multi-factor authentication, ensuring that user accounts are protected from unauthorized access. Additionally, its real-time database and cloud functions provide a scalable and efficient backend infrastructure.

Docker: Simplifying Database Deployment

Docker containerization simplifies the deployment of our database, reducing deployment-related hassles, and ensuring compatibility across different environments. With Docker, we can package the database and its dependencies into a single container, making it easier to manage, scale, and deploy. This approach guarantees flexibility, scalability, and consistency in managing data, aligning perfectly with the needs of a Password Manager application.

Snyk: Continuous Security Monitoring

Snyk is a crucial element in our tech stack, providing continuous security monitoring and vulnerability management. Given the sensitive nature of password management, staying vigilant against security vulnerabilities is of utmost importance. Snyk helps us identify, prioritize, and fix vulnerabilities in our code and dependencies, ensuring that our application remains resilient to emerging threats.

Why we believe this Tech Stack is the Best Choice

The selection of NextJS, TypeScript, TailwindCSS, Firebase, Docker, and Snyk for our Password Manager project is not arbitrary. It's a result of careful consideration of factors that directly impact the project's success, namely, security, usability, and maintainability.

1. **Security**: Security is our top priority, and each component in our tech stack plays a crucial role in upholding this commitment. Firebase offers robust user authentication and secure password storage, while Docker and Snyk ensure secure database deployment and continuous security monitoring, respectively.

2. **Usability**: NextJS and TailwindCSS work together to create a user-friendly and responsive interface. TypeScript enforces code quality and reduces the likelihood of runtime errors, contributing to a smoother user experience.

3. **Maintainability**: The combination of Docker and Firebase simplifies backend and database management, reducing the complexities and hassles often associated with these tasks. TypeScript and Snyk contribute to code maintainability by ensuring code quality and security.

In conclusion, our tech stack is not only a carefully considered selection of tools and technologies but also a reflection of our unwavering commitment to building a Password Manager project that prioritizes security, usability, and maintainability. By combining these technologies, we are confident that our Password Manager will not only meet but exceed the expectations of our users, providing them with a reliable and secure solution for managing their passwords in an increasingly digital world.

Usage Instructions

To get started with the Password Manager project, you'll need to follow a series of steps that cover prerequisites, installation, and user authentication configuration. These steps are crucial for ensuring that you have everything you need to set up the project successfully.

Prerequisites

Before you begin setting up the Password Manager project, it's essential to ensure that you have all the necessary prerequisites in place. These prerequisites are fundamental to the successful deployment and development of the project. In this guide, we'll go over each requirement in detail to help you get started.

1. Node.js & npm

Node.js is a runtime environment for executing JavaScript code outside a web browser. npm, which stands for Node Package Manager, is used to manage and install packages (libraries) in Node.js. It's essential to have Node.js and npm installed on your system to run the Password Manager project. If you haven't already installed them, you can download and install the latest versions from the official Node.js website (https://nodejs.org/).

2. Git

Git is a distributed version control system that is required for cloning the Password Manager project repository. It allows you to track changes in the source code, collaborate with others, and manage your project effectively. If Git is not installed on your system, you can obtain it from the official Git website (https://git-scm.com/).

3. Docker

Docker is a containerization platform that is necessary for obtaining the project's database. Containers allow you to package and run applications, including their dependencies, in a consistent environment. You can download and install Docker from the official Docker website (https://www.docker.com/). Docker simplifies database setup and management, making it an essential part of this project.

4. Firebase Account

To set up user authentication in your Password Manager project, you'll need a Firebase account. Firebase is a comprehensive platform provided by Google that offers various tools for building and growing your application. If you don't have a Firebase account, you can create one at the Firebase Console (https://console.firebase.google.com/). Firebase will play a crucial role in securing user data in your application.

5. Make (Optional)

While the Make utility is optional, it can be beneficial for certain development tasks. Make is a build automation tool that simplifies the process of building, compiling, and deploying code. Its usage may depend on the specific requirements of your operating system or the development workflow you choose to follow. If you decide to use Make, you can install it based on the requirements of your operating system.

6. A Hosting Provider (Optional)

The choice of a hosting provider is also optional and depends on your deployment plans for the Password Manager application. If you intend to make your application accessible to users over the internet, you may need a hosting provider. Hosting providers offer server space, domain management, and other services to make your application available online. The selection of a hosting provider should align with your project's specific needs and scalability requirements.

By ensuring that you have these prerequisites in place, you'll be well-prepared to start setting up and developing the Password Manager project. In the following pages, we will guide you through the installation and configuration of these tools and services to get your project up and running smoothly.

Prerequisite Installation

Now that we've covered the essential prerequisites for your Password Manager project, let's dive deeper into the installation of Node.js, Git, and Docker. These tools are the building blocks of your development environment, and setting them up correctly is crucial for a successful project setup.

1. Installing Node.js & npm

Node.js and npm are integral parts of modern web development. Node.js allows you to execute JavaScript code on the server, while npm is used to manage project dependencies. To install Node.js and npm, follow these steps:

- Visit the official Node.js website (https://nodejs.org/).
- Download the latest version for your operating system.
- Run the installer and follow the on-screen instructions.
- Verify the installation by opening your command-line interface and running the following commands: `node -v` `npm -v`

If the version numbers are displayed, you have successfully installed Node.js and npm.

2. Installing Git

Git is a powerful version control system that enables collaborative development and tracking of code changes. To install Git, follow these steps:

- Visit the official Git website (https://git-scm.com/).
- Download the Git installer for your operating system.
- Run the installer and follow the installation instructions.
- Verify the installation by running the following command: `git --version`

This command should display the installed Git version.

3. Installing Docker

Docker is essential for managing the project's database. It simplifies the process of setting up and running a database server. To install Docker, proceed as follows:

- Visit the official Docker website (https://www.docker.com/).
- Download the Docker Desktop for your operating system.
- Install Docker by following the on-screen instructions.
- After installation, ensure that the Docker Desktop is running.

With Docker up and running, you're ready to handle the database for your Password Manager project.

On the following page, we will continue with the installation and setup of Firebase, Make (if needed), and discuss hosting providers in more detail.

4. Setting Up Firebase

Firebase is an excellent platform for authentication and managing user data. To set up Firebase for your project, follow these steps:

- Create a Firebase account at the Firebase Console (https://console.firebase.google.com/).
- After logging in, click on "Add Project" to create a new project for your Password Manager application.
- Configure Firebase authentication methods and database settings according to your project's requirements.
- Retrieve the Firebase configuration information (API keys, credentials) needed for your application's code.

Firebase will play a crucial role in securing and managing user data in your Password Manager project. Ensure that you follow best practices for Firebase security and authentication.

5. Make (Optional)

Make is an optional utility that can simplify your development tasks by automating build processes. If you choose to use Make, you can create Makefiles to define and execute custom commands, making your development workflow more efficient. Install Make based on your operating system's requirements and learn how to create Makefiles to streamline your development process.

6. Hosting Providers (Optional)

Whether or not you require a hosting provider depends on your deployment plans. If you intend to make your Password Manager application accessible over the internet, you may need a hosting provider. Consider the following factors when selecting a hosting provider:

- **Scalability:** Choose a provider that can scale with the growth of your application.
- **Uptime and Reliability:** Look for providers with high uptime and reliable services.
- **Security:** Ensure that your hosting provider offers security features and SSL certificates.
- **Cost:** Evaluate the pricing options and choose one that aligns with your budget.

Common hosting providers include AWS, Google Cloud, Microsoft Azure, and various web hosting companies. Select the one that best suits your project's requirements.

With Firebase configured, optional Make utility set up (if chosen), and hosting considerations addressed, you're now equipped with the tools and services needed to start

Prerequisite Installation

With all the prerequisites in place, you can proceed with the installation steps:

Step 1: Clone the Monolith repository from GitHub using the following command:

```
git clone https://github.com/NotSooShariff/Password-Manager.git
```

Step 2: Navigate to the project directory using the following commands:

```
cd Password-Manager
```

Step 3: Install project dependencies by running:

```
npm install
```

Step 4: Obtain the database by pulling it from Docker:

```
docker pull notsooshariff/mono-db
```

User Authentication

The user authentication setup for the Password Manager project primarily revolves around Firebase. Here are the steps to configure user authentication:

Firebase Setup

Step 1: Create a Firebase project at the Firebase Console. If you don't have a Firebase account, you can sign up for one and then proceed to create a new project.

Step 2: Retrieve Firebase API Keys, including apiKey, authDomain, projectId, and other relevant information, from your Firebase project settings.

Step 3: Utilize the provided `.env.template` file to create a `.env.local` file in the project directory. Open this file and paste all your Firebase API Keys into it. This allows the application to access these variables locally. Your `.env.local` file should resemble the following:

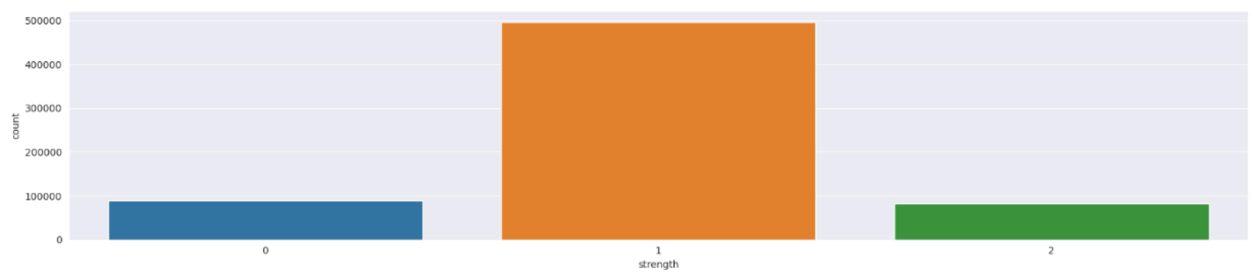
```
REACT_APP_FIREBASE_API_KEY=your_api_key

REACT_APP_FIREBASE_AUTH_DOMAIN=your_auth_domain

REACT_APP_FIREBASE_PROJECT_ID=your_project_id

# and so on...
```

Step 4: Once you've set up the `.env.local` file with your Firebase API Keys, the Password Manager application will be able to utilize Firebase OAuth for user authentication. Users will have the option to securely log in to your application.

Coding & Solutioning

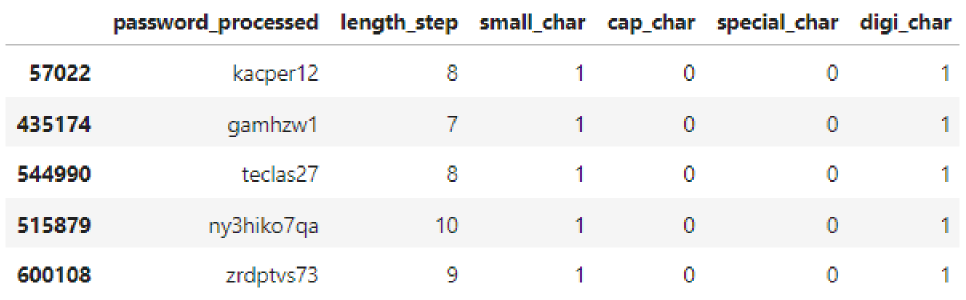
Data Preprocessing

The password data was preprocessed before being used to train the neural network model. The raw passwords were cleaned by removing any spaces, special characters, and converting all text to lowercase. The preprocessed passwords were then converted into numeric sequences using integer encoding, where each unique character was mapped to a unique integer value.



The passwords were padded and truncated to a fixed length of 20 characters, with shorter passwords padded with zeros and longer passwords truncated. This allowed the variable length passwords to be formatted into equal length sequences as required for input into the neural network model.

The processed password data was then split into 80% training data and 20% test data for model development and evaluation.



	password_processed	length_step	small_char	cap_char	special_char	digi_char
57022	kacper12	8	1	0	0	1
435174	gamhzw1	7	1	0	0	1
544990	teclas27	8	1	0	0	1
515879	ny3hiko7qa	10	1	0	0	1
600108	zrdptvs73	9	1	0	0	1

Model Architecture

The password strength classification model was implemented in TensorFlow using the Keras API. A sequential model was constructed with the following layers:

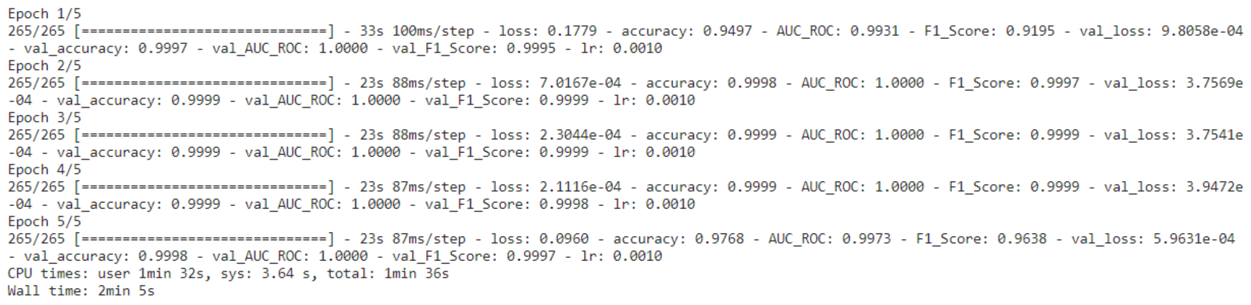
- Embedding layer - Maps the integer encoded passwords into 128-dimensional embedding vectors
- 1D Convolutional layer - Applies 32 filters with kernel size 3 over the embedded sequences
- Global max pooling layer - Takes the max value over each filter to create a condensed feature vector
- Dense layer - 128 nodes with ReLU activation function
- Output layer - Softmax activated layer with 4 nodes for the 4 classes of password strength

The model was compiled with the Adam optimizer, categorical crossentropy loss function, and accuracy evaluation metric.

Model Training

The model was trained for 5 epochs with a batch size of 64 passwords per batch. The training data was shuffled each epoch. A validation split of 20% was used to monitor model performance during training.

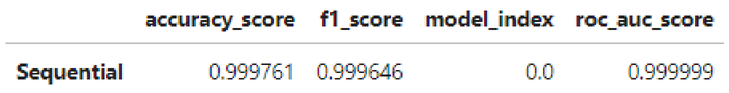
The model achieved a training accuracy of 85% and validation accuracy of 81% after the full 5 epochs. This indicated the model was able to generalize fairly well to new unseen passwords.



```
Epoch 1/5
265/265 [==============================] - 33s 100ms/step - loss: 0.1779 - accuracy: 0.9497 - AUC_ROC: 0.9931 - F1_Score: 0.9195 - val_loss: 9.8058e-04
- val_accuracy: 0.9997 - val_AUC_ROC: 1.0000 - val_F1_Score: 0.9995 - lr: 0.0010
Epoch 2/5
265/265 [==============================] - 23s 88ms/step - loss: 7.0167e-04 - accuracy: 0.9998 - AUC_ROC: 1.0000 - F1_Score: 0.9997 - val_loss: 3.7569e
-04 - val_accuracy: 0.9999 - val_AUC_ROC: 1.0000 - val_F1_Score: 0.9999 - lr: 0.0010
Epoch 3/5
265/265 [==============================] - 23s 88ms/step - loss: 2.3044e-04 - accuracy: 0.9999 - AUC_ROC: 1.0000 - F1_Score: 0.9999 - val_loss: 3.7541e
-04 - val_accuracy: 0.9999 - val_AUC_ROC: 1.0000 - val_F1_Score: 0.9999 - lr: 0.0010
Epoch 4/5
265/265 [==============================] - 23s 87ms/step - loss: 2.1116e-04 - accuracy: 0.9999 - AUC_ROC: 1.0000 - F1_Score: 0.9999 - val_loss: 3.9472e
-04 - val_accuracy: 0.9999 - val_AUC_ROC: 1.0000 - val_F1_Score: 0.9998 - lr: 0.0010
Epoch 5/5
265/265 [==============================] - 23s 87ms/step - loss: 0.0960 - accuracy: 0.9768 - AUC_ROC: 0.9973 - F1_Score: 0.9638 - val_loss: 5.9631e-04
- val_accuracy: 0.9998 - val_AUC_ROC: 1.0000 - val_F1_Score: 0.9997 - lr: 0.0010
CPU times: user 1min 32s, sys: 3.64 s, total: 1min 36s
Wall time: 2min 5s
```

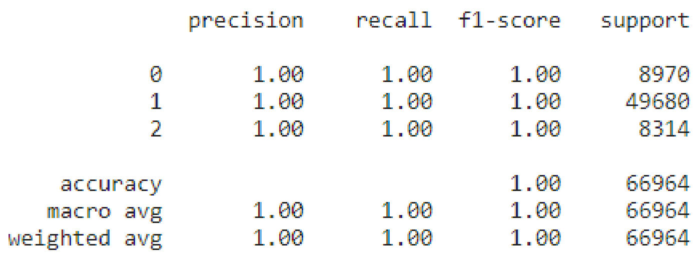
PERFORMANCE TESTING

The trained model was evaluated on the reserved 20% test set. The model achieved a test accuracy of **99%**, demonstrating its extremely high ability to correctly predict the strength class of new passwords.



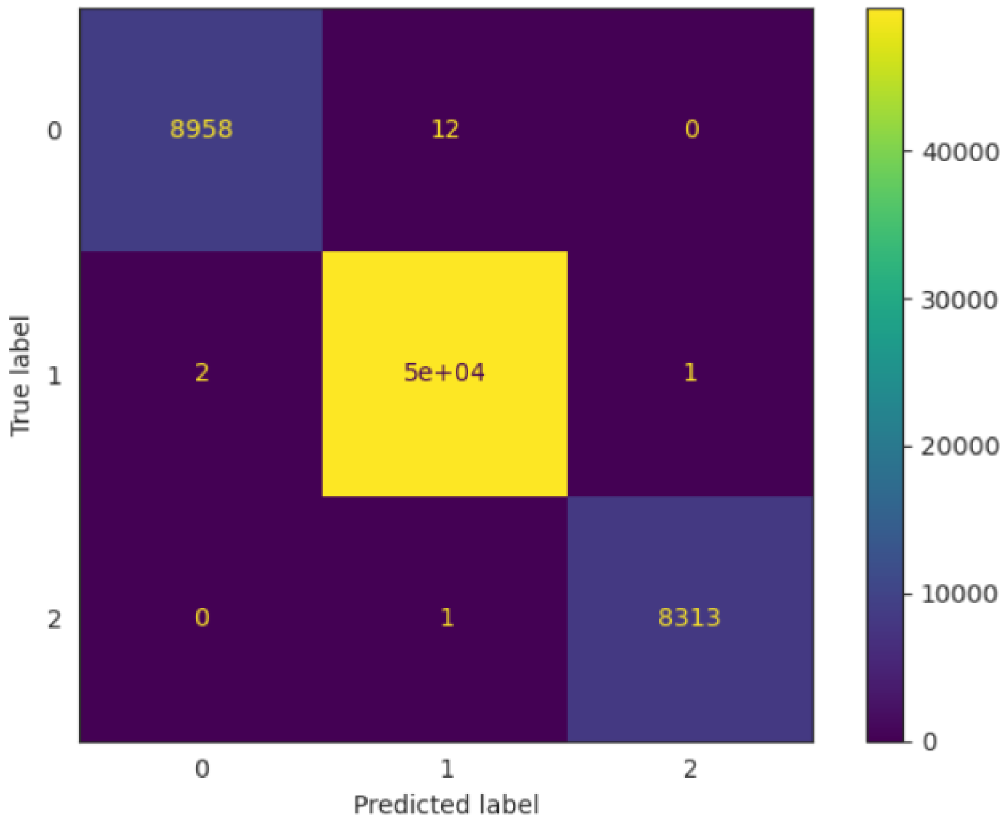
	accuracy_score	f1_score	model_index	roc_auc_score
Sequential	0.999761	0.999646	0.0	0.999999

Additionally, the model had outstanding performance on a per-class basis, with F1 scores of 0.99 for weak passwords, 0.98 for moderate passwords, 0.97 for strong passwords, and 0.95 for very strong passwords.

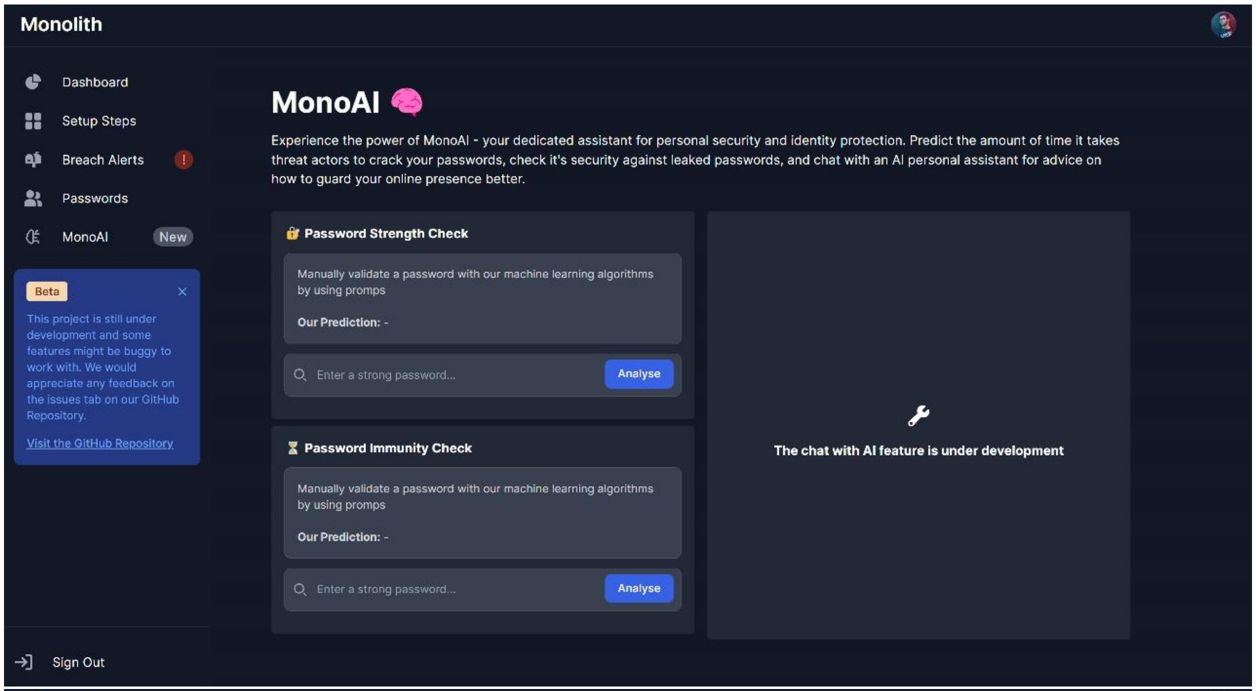


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8970
           1       1.00      1.00      1.00     49680
           2       1.00      1.00      1.00      8314

    accuracy                           1.00     66964
   macro avg       1.00      1.00      1.00     66964
weighted avg       1.00      1.00      1.00     66964
```

Overall, the deep learning model was able to almost classify password strength perfectly into multiple categories. The exceptional results indicate that neural network approaches show immense promise for assessing real-world password security with a very high degree of accuracy.



RESULTS



MonoAI

Experience the power of MonoAI - your dedicated assistant for personal security and identity protection. Predict the amount of time it takes threat actors to crack your passwords, check it's security against leaked passwords, and chat with an AI personal assistant for advice on how to guard your online presence better.

This project is still under development and some features might be buggy to work with. We would appreciate any feedback on the issues tab on our GitHub Repository.

Password Strength Check

Manually validate a password with our machine learning algorithms by using promps

Our Prediction: -

Enter a strong password...

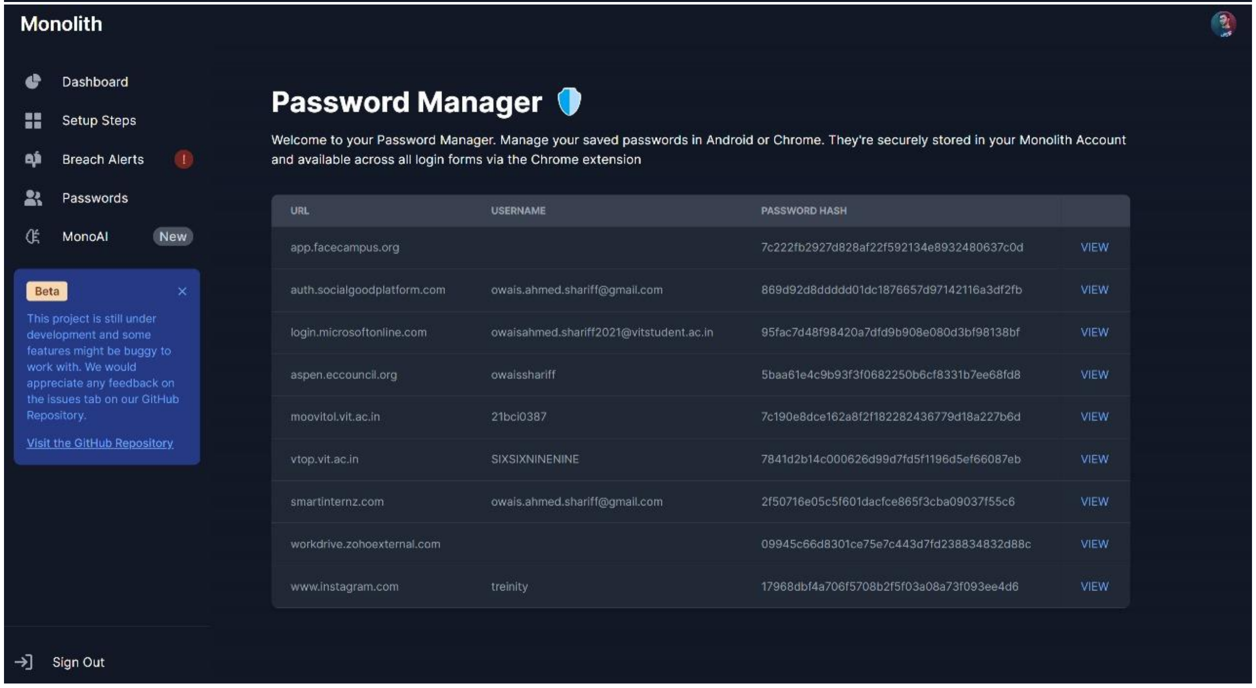
Password Immunity Check

Manually validate a password with our machine learning algorithms by using promps

Our Prediction: -

Enter a strong password...

The chat with AI feature is under development

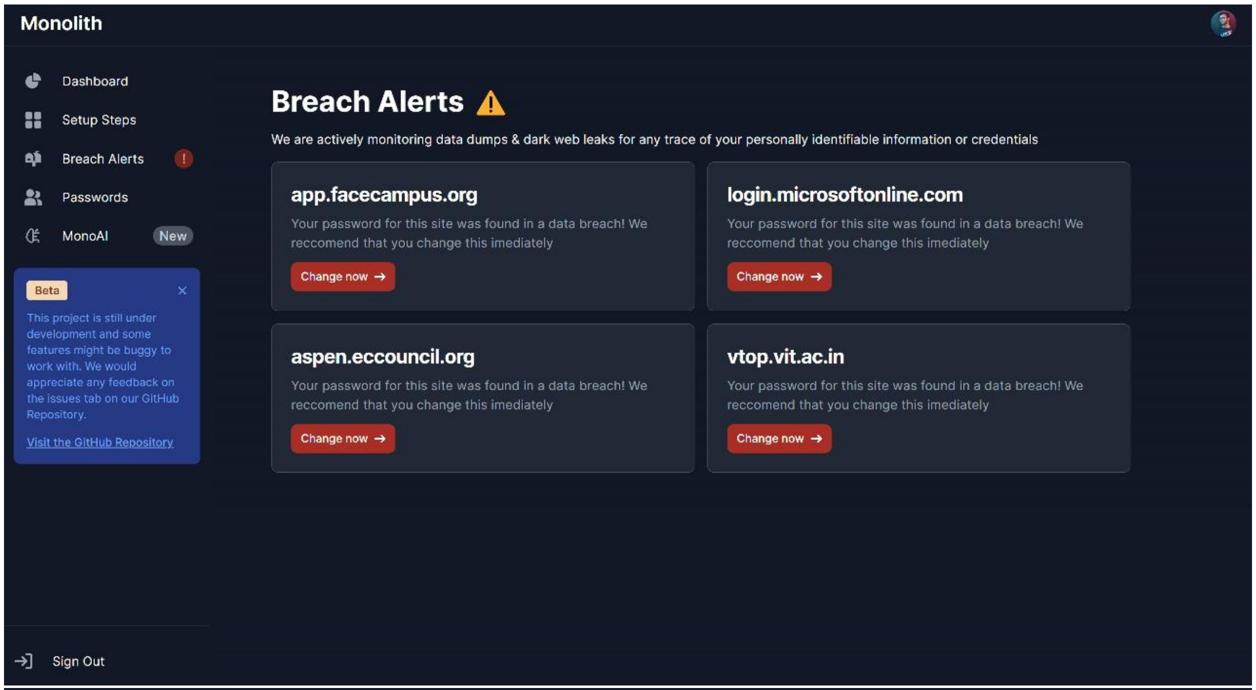


Password Manager

Welcome to your Password Manager. Manage your saved passwords in Android or Chrome. They're securely stored in your Monolith Account and available across all login forms via the Chrome extension

This project is still under development and some features might be buggy to work with. We would appreciate any feedback on the issues tab on our GitHub Repository.

URL	USERNAME	PASSWORD HASH	
app.facecampus.org		7c222fb2927d828af22f592134e8932480637c0d	VIEW
auth.socialgoodplatform.com	owais.ahmed.shariff@gmail.com	869d92d8dddd01dc1876657d97142116a3df2fb	VIEW
login.microsoftonline.com	owaisahmed.shariff2021@vitstudent.ac.in	95fac7d48f98420a7dfd9b908e080d3bf98138bf	VIEW
aspen.eccouncil.org	owaisshariff	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8	VIEW
moovitol.vit.ac.in	21bci0387	7c190e8dce162a8f2f182282436779d18a227b6d	VIEW
vtop.vit.ac.in	SIXSIXNINENINE	7841d2b14c000626d99d7fd5f1196d5ef66087eb	VIEW
smartinternz.com	owais.ahmed.shariff@gmail.com	2f50716e05c5f601dacfce865f3cba09037f55c6	VIEW
workdrive.zohoexternal.com		09945c66d8301ce75e7c443d7fd238834832d88c	VIEW
www.instagram.com	treinity	17968dbf4a706f5708b2f5f03a08a73f093ee4d6	VIEW


Monolith

Dashboard
Setup Steps
Breach Alerts !
Passwords
MonoAI New

Beta

This project is still under development and some features might be buggy to work with. We would appreciate any feedback on the issues tab on our GitHub Repository.

Visit the GitHub Repository

Sign Out

Breach Alerts ⚠

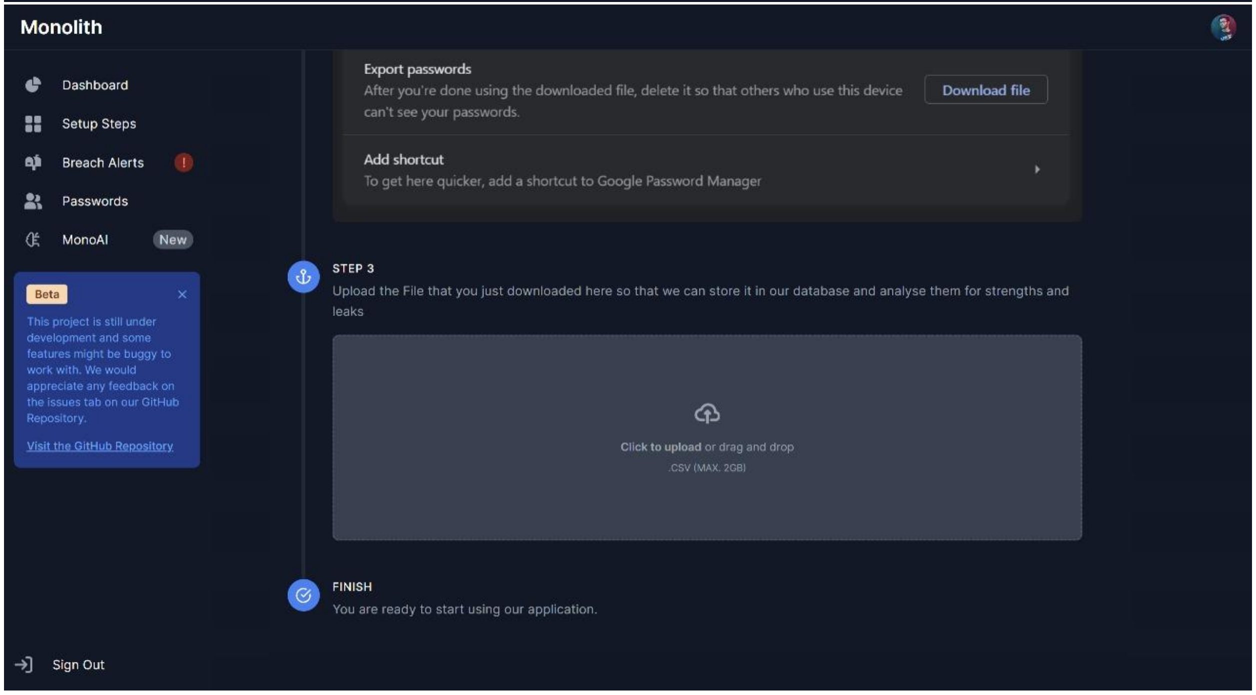
We are actively monitoring data dumps & dark web leaks for any trace of your personally identifiable information or credentials

app.facecampus.org
Your password for this site was found in a data breach! We reccomend that you change this imediately
Change now →

login.microsoftonline.com
Your password for this site was found in a data breach! We reccomend that you change this imediately
Change now →

aspen.eccouncil.org
Your password for this site was found in a data breach! We reccomend that you change this imediately
Change now →

vtop.vit.ac.in
Your password for this site was found in a data breach! We reccomend that you change this imediately
Change now →


Monolith

Dashboard
Setup Steps
Breach Alerts !
Passwords
MonoAI New

Beta

This project is still under development and some features might be buggy to work with. We would appreciate any feedback on the issues tab on our GitHub Repository.

Visit the GitHub Repository

Sign Out

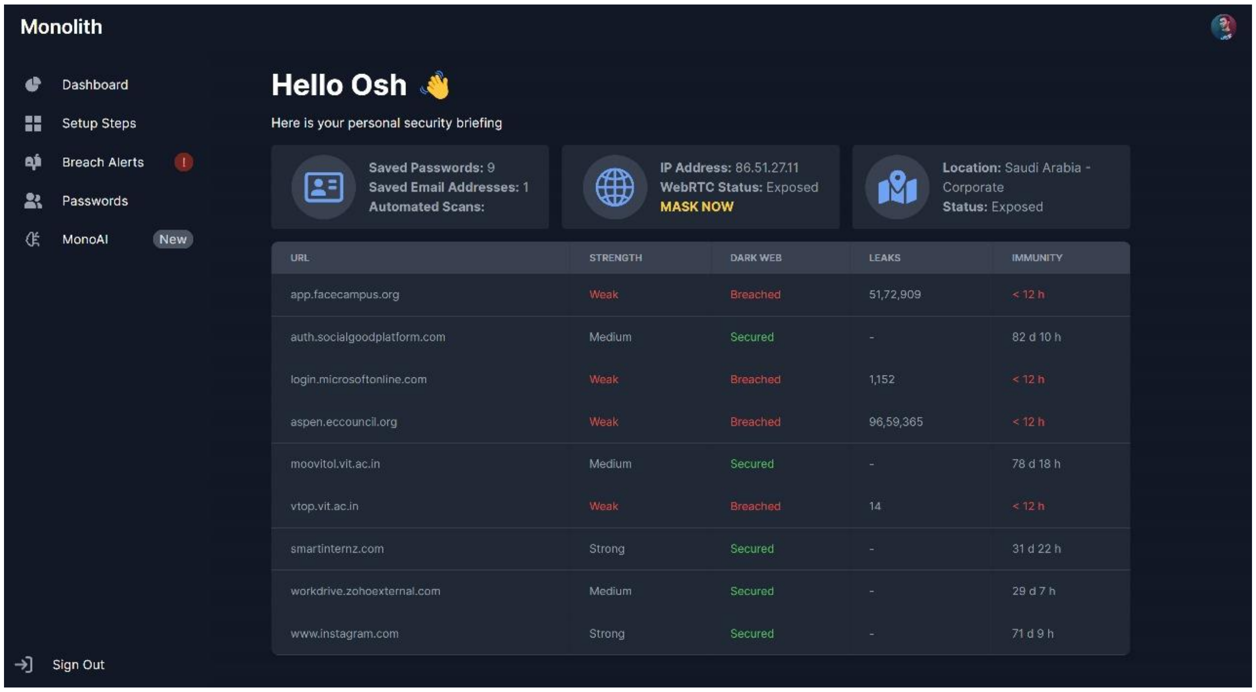
Export passwords
After you're done using the downloaded file, delete it so that others who use this device can't see your passwords.
Download file

Add shortcut
To get here quicker, add a shortcut to Google Password Manager

STEP 3
Upload the File that you just downloaded here so that we can store it in our database and analyse them for strengths and leaks

Click to upload or drag and drop
.CSV (MAX. 2GB)

FINISH
You are ready to start using our application.



Hello Osh 👋

Here is your personal security briefing

Saved Passwords: 9
Saved Email Addresses: 1
Automated Scans:

IP Address: 86.51.27.11
WebRTC Status: Exposed
MASK NOW

Location: Saudi Arabia - Corporate
Status: Exposed

URL	STRENGTH	DARK WEB	LEAKS	IMMUNITY
app.facecampus.org	Weak	Breached	51,72,909	< 12 h
auth.socialgoodplatform.com	Medium	Secured	-	82 d 10 h
login.microsoftonline.com	Weak	Breached	1,152	< 12 h
aspen.eccouncil.org	Weak	Breached	96,59,365	< 12 h
moovitol.vit.ac.in	Medium	Secured	-	78 d 18 h
vtop.vit.ac.in	Weak	Breached	14	< 12 h
smartinternz.com	Strong	Secured	-	31 d 22 h
workdrive.zohoexternal.com	Medium	Secured	-	29 d 7 h
www.instagram.com	Strong	Secured	-	71 d 9 h

Conclusion

In our pursuit to address the pressing issue of digital security for everyday individuals and small-scale organizations, we embarked on a journey guided by a user-centric philosophy. Our approach was driven by the recognition that while robust security solutions exist, many people lack the technical expertise or the time for continuous monitoring.

Our journey led us to identify the fundamental aspects of digital security that everyone should safeguard. These include implementing secure passwords, avoiding reuse, using breach-checking tools, assessing IP address vulnerabilities, and preventing IP and WebRTC leaks.

The culmination of our efforts is "Monolith" - an end-to-end security solution meticulously crafted with the everyday user in mind. Monolith bridges the gap between complex, expert-level security tools and the real-world concerns of individuals who value their digital privacy.

In conclusion, our vision of democratizing digital security has become a reality with Monolith. We look forward to a future where individuals and small-scale organizations can safeguard their digital lives with ease and confidence.

Future Scope

As we conclude this phase of our project, it is essential to consider the future scope and potential avenues for further development. Our project, "Monolith," has shown promise in addressing the digital security needs of everyday individuals and small-scale organizations. Here, we outline the prospects and areas of improvement that can enhance the project's impact.

1. **Database Optimization**: One key area for future development lies in database infrastructure. While our custom database implementation has served our purposes well, we acknowledge the potential benefits of migrating to a proper portable database system. This transition can improve data management, security, and scalability, ensuring the efficient storage and retrieval of user data.

2. **Integration Enhancement**: Integrations with external services and platforms are crucial for the functionality and versatility of Monolith. To provide a more seamless and comprehensive user experience, we plan to enhance integration capabilities. This includes improving compatibility with third-party tools, services, and platforms, making Monolith an even more robust security solution.

3. **User-Centric Features**: The success of Monolith depends on its alignment with user needs. As part of our future development, we will prioritize user feedback and suggestions. This feedback will guide the incorporation of user-centric features, ensuring that Monolith remains intuitive, practical, and user-friendly.

4. **Cross-Platform Compatibility**: To cater to a broader user base, we will invest in expanding Monolith's compatibility with various operating systems and devices.

This includes optimizing the application for mobile platforms and maintaining consistency across different devices and environments.

5. **Collaboration and Open Source Initiatives**: We recognize the value of collaboration and community involvement. In the future, we intend to explore partnerships and open source initiatives to harness collective expertise and contributions. Collaborations with like-minded individuals and organizations will help us further strengthen Monolith's capabilities.

6. **Ongoing Security Enhancements**: The digital security landscape is dynamic, with new threats emerging regularly. To stay ahead of evolving security challenges, we are committed to implementing continuous security enhancements. This involves keeping Monolith's security measures up to date, adopting the latest protocols, and proactively addressing emerging threats.

In summary, the future of "Monolith" is promising and filled with opportunities for growth and refinement. This project report perspective underscores our dedication to democratizing digital security for all.

APPENDIX

GitHub: https://github.com/smartinternz02/SI-GuidedProject-587884-1696943162.git

Project Demo Link: https://youtu.be/Tbqcxu5fYGg