Guardian Monitor App Security and Privacy Policy

The Guardian Monitor App is designed to monitor and manage various activities, providing users with a secure and private experience. This document outlines the measures taken to ensure data security and privacy, aligning with industry standards and regulatory requirements

FOR USERS

1 Data Collection:

1.1. Types of Data Collected:

- **Personal Information:** This includes basic details such as your full name, email address, and contact number. We need this information to create your account and manage communication with you, such as sending notifications and customer support.
- **Activity Data:** The app tracks your interaction with different features, including logs of actions you perform within the app (e.g., when you log in, what features you access). This helps us monitor app performance and provide personalized insights.
- **Device Information:** We gather technical information from your device, such as its IP address, operating system, and device type. This data is used for compatibility purposes and to ensure smooth functioning across different devices.
- **Location Data:** If you use location-based features in the app, we may collect precise geolocation data. This is optional and only collected with your consent.

1.2. Why We Collect Data?

- **App Functionality**: Your personal data is essential for the app's core functions, such as creating and maintaining your account, and making the app's features work as intended.
- **User Experience Personalization:** We use your data to offer tailored experiences, like recommendations or reminders that are relevant to how you use the app.
- **Legal Compliance**: In some cases, we are required by law to collect and store certain types of data (e.g., to comply with court orders or legal investigations).

2. Data Storage:

2.1. Encryption:

We use Advanced Encryption Standard (AES256) to secure your data when it's stored on our servers. For data transferred between your device and our servers, we employ

Transport Layer Security (TLS 1.2+) to ensure that it remains private and secure during transmission ("in transit"). This makes it extremely difficult for anyone to intercept or read your data without authorization.

2.2. Data Retention:

We retain your data only as long as necessary to fulfill the purposes outlined in this policy or as required by law. For example, we keep your account details while your account is active and delete them if you request account termination (except in cases where we are legally obliged to keep certain information).

You have the right to request the deletion of your personal data, and we will comply unless there are legal reasons to retain it.

2.3. Access Control:

Role Based Access Control (RBAC): Only authorized personnel, based on their role within the organization, can access sensitive data. This minimizes the risk of unauthorized access or misuse of data.

MultiFactor Authentication (MFA): All administrative or privileged access to your data requires MFA, meaning that logging in requires not just a password, but also a secondary verification method, like a code sent to a mobile device.

3. Data Sharing and Disclosure:

3.1 Third-Party Service Providers

Data may be shared with third-party service providers who assist in the app's operation (e.g., cloud hosting, analytics). These providers are required to adhere to strict data security and privacy standards.

3.2. Legal Requirements

Data may be disclosed to comply with legal obligations, such as responding to court orders or legal processes.

3.3. User Consent

Data sharing with third parties for purposes other than those mentioned above will require explicit user consent.

4. Your Rights:

4.1. Access to Data:

You can request access to the data that the Guardian Monitor App has collected about you. Upon request, we will provide you with a detailed record of the information we hold about you in a secure format.

4.2. Data Correction:

If you notice any inaccuracies in the personal data we have on file, you can request to have this information corrected. This ensures that the data we use for your app experience is always accurate and up to date.

4.3. Data Deletion:

You have the right to ask for your data to be deleted (also known as the "right to be forgotten"). Once requested, we will remove your personal information from our systems unless we have a legitimate legal reason to retain it (e.g., pending litigation).

4.4. Opt-out of Certain Features:

You can adjust your app settings to limit the types of data we collect (e.g., opting out of location tracking or certain types of personalized services). However, some features may not work as intended if data collection is limited.

5. Security Measures:

5.1. Regular Security Audits:

We regularly conduct security audits to identify vulnerabilities in the system and implement necessary updates. These audits are performed by security experts to ensure compliance with the latest security standards.

5.2. Incident Response Plan:

In case of a data breach or any security incident, we have an incident response plan to immediately contain the threat, assess its impact, and inform affected users. If necessary, we will notify you promptly and work with the relevant authorities to address the issue.

5.3. User Education and Protection:

We believe that security is a shared responsibility. To this end, we offer guidance on how to protect their accounts, such as setting strong passwords, enabling MFA, and recognizing phishing attempts. These resources are available in-app and on our website.

6. Policy Updates:

Our privacy and security policies are regularly reviewed and updated to reflect changes in technology, regulatory requirements, and app features. Any significant changes will be communicated to you via email or through notifications within the app. We recommend reviewing the updated policy to stay informed.

7. Contact Information:

If you have any questions or concerns regarding your data privacy or this policy, you can reach us at support@guardianmonitor.com. We are here to help address any queries or resolve any issues you might have regarding your privacy and data security.

For Developers:

1. Data Collection:

1.1. Types of Data Collected:

Developers should implement mechanisms for collecting the following data types in line with security best practices:

- Personal Information: Collect basic user details like name and email during account creation. Ensure secure storage and limit unnecessary personal data collection to what is needed for core functionality.
- Activity Data: Capture app interactions and logs to monitor performance and provide user insights. Anonymize this data wherever possible, especially when used for analytics.
- Device Information: Collect IP addresses, device types, and OS versions for diagnostic and compatibility purposes. Implement logging for troubleshooting while minimizing personally identifiable information (PII) exposure.
- Location Data: Only collect location data when required for specific app features. Ensure that users must give explicit consent before tracking their location, and always provide an option to optout.

1.2. Purpose of Data Collection:

Optimize user experience, ensuring that data is collected to improve performance, personalize features, and comply with legal requirements.

Implement checks to ensure that any data collection aligns with Australian privacy laws and provides users with transparency about why and how their data is being used.

2. Data Storage:

2.1. Encryption:

At Rest: Sensitive data, including user credentials and PII, must be encrypted using AES256. Ensure encryption keys are managed securely, and implement key rotation policies.

In Transit: All data transfers between client devices and the server should be encrypted using TLS 1.2+. Verify the certificates and maintain HTTPS on all connections to prevent man in the middle attacks.

2.2. Access Control:

Implement Role Based Access Control (RBAC): Define roles that restrict access to sensitive data, limiting access to only those who need it.

MFA: Ensure that all staff and administrators accessing sensitive systems are required to authenticate using MultiFactor Authentication (MFA). This adds an additional layer of security by requiring a second form of authentication, such as a onetime password (OTP).

3. Data Sharing and Disclosure:

3.1. Third-party Service Providers:

Before integrating any third-party service provider (e.g., for cloud hosting, analytics, or payments), ensure they meet the app's security requirements. Only work with providers that adhere to Data Processing Agreements (DPAs) to protect user information.

Regularly audit third-party services to ensure compliance with contractual obligations and industry standards.

3.2. User Consent for Additional Sharing:

Implement user consent management frameworks within the app. If any data is to be shared for marketing or nonoperational purposes, ensure explicit consent is gathered, logged, and can be revoked easily by the user.

4. User Rights Management:

4.1. Data Access Requests:

Develop API endpoints to provide users with access to their data. Ensure the data is provided in a secure and encrypted format, and that proper logging is in place for when users request their data.

4.2. Data Correction:

Build features allowing users to correct any inaccurate data directly within the app. Ensure that all changes are logged and that corrections do not introduce data integrity issues.

4.3. Data Deletion:

Implement secure deletion mechanisms that comply with Australian regulations. When users request data deletion, ensure that personal data is purged both from primary databases and backups. Implement deletion logging for auditing purposes.

4.4. Opt-out of Features:

Allow users to optout of features that require sensitive data collection, such as location tracking or analytics. Ensure that these settings are easily accessible and communicated clearly to users.

5. Security Measures:

5.1. Regular Security Audits:

Schedule regular security audits to ensure that the app adheres to the latest security standards. Vulnerabilities identified in the audits should be addressed and patched in a timely manner.

5.2. Incident Response Plan:

Develop a comprehensive incident response plan (IRP) that outlines procedures for identifying, responding to, and recovering from security breaches. Regularly test and update the IRP to cover new threat scenarios.

5.3. User Education and Protection:

Provide users with security best practices through inapp resources, emails, and notifications. This includes encouraging the use of strong passwords, enabling MFA, and being aware of phishing attacks.

6. Policy Updates:

Maintain a version-controlled policy update process to reflect changes in legal requirements or app features. Ensure that significant updates are reviewed by legal experts and communicated clearly to both users and stakeholders.

Compliance with Regulation according to Australian Laws

In Australia, healthcare apps must comply with a variety of privacy and security laws and regulations to protect sensitive health data. The primary legislative framework includes:

(a) **Privacy Act 1988:** The foundation of Australia's privacy laws, which are applicable to businesses in the private sector, is this. It establishes guidelines for the collection, use, disclosure, and security of personal information, including health data. Thirteen Australian Privacy Principles (APPs) that establish guidelines for handling personal data are included in the Privacy Act. These include:
APPs 1 and 2: Demanding that personal data be managed in an open and transparent manner and giving people the choice to withhold their identity.
APP 11: Requires companies to take appropriate measures to safeguard personal information against loss, unlawful access, alteration, and abuse.

(b) **My Health Records Act 2012:** This is the policy that oversees the national electronic health record system, My Health Record. Individuals have control over who has access to their health information thanks to the strict laws it incorporates regarding the security, privacy, and confidentiality of health data.

(c) **Healthcare Identifiers Act 2010**: To guarantee proper identification of people inside the healthcare system, this act governs the use of healthcare identifiers, which are unique IDs provided to people, healthcare professionals, and organizations.

(d) **Notifiable Data Breaches (NDB) Scheme (2018)**: This is a requirement of the Privacy Act that states that when a data breach involving personal information is likely to cause significant harm, the affected persons and the Office of the Australian Information Commissioner (OAIC) must be notified.

(e) **Therapeutic Goods Act (TG Act):** To ensure that therapeutic items satisfy safety and efficacy criteria, this act governs them, including software that serves as a medical device. A healthcare app must abide by the Therapeutic Goods Act and be registered on the Australian Register of Therapeutic Goods (ARTG) to be considered a medical device.

(f) **Australian Cyber Security Centre (ACSC) Guidelines**: To protect yourself from online dangers, go by the ACSC's security recommendations. To safeguard the private information the app manages, this entails putting strong encryption, safe authentication procedures, frequent security audits, and effective incident response strategies into place.

Since the Guardian Monitor app handles sensitive patient data, compliance with Australian privacy and security legislation is essential. Respecting these rules not only saves patients' rights but also shields the app from future legal issues and improves its reputation in the medical community. To be compliant when laws change, it is advised to have regular legal consultations and updates.

Implementation of Security and Privacy for App

1. Data Encryption and Storage:

Implementation: For secure data transmission and storage, use TLS (Transport Layer Security) 1.2+ for all API communications between the Android client and the Node.js server. On the server side, ensure that sensitive data like personal user information and activity logs are encrypted using AES-256 encryption before being stored in the database.

Example: Google Fit encrypts health-related data both in transit and at rest to comply with healthcare regulations.

Node.js Example: You can use libraries like crypto or bcrypt to encrypt sensitive data. For TLS, ensure your Node.js API server uses HTTPS with certificates from trusted providers like Let's Encrypt.

Code Snippet eg:

```
const crypto = require('crypto');

const secret = 'your_secret_key';

const hash = crypto.createHmac('sha256', secret)

        .update('Sensitive Data')

        .digest('hex');
```

Different encryption methods can include:

1. **RSA (Asymmetric Encryption):** RSA is an asymmetric encryption algorithm often used for securing sensitive data transmissions. You can encrypt data with a public key and decrypt it with a private key.
2. **bcrypt for Password Hashing:** While not strictly encryption, bcrypt is used to hash passwords securely. The idea is to hash passwords before storing them in the database and never decrypt them.
3. **Elliptic Curve Cryptography (ECC):** ECC is an alternative to RSA, which offers strong encryption with smaller keys. You can use the elliptic package in Node.js for ECC.
4. **ChaCha20:** It is a modern and fast stream cipher that offers excellent security. It's suitable for environments where speed is critical. You can use the crypto module in Node.js for ChaCha20 as well.

2. Role-Based Access Control (RBAC) and Multi-Factor Authentication (MFA):

Implementation: Implement RBAC to restrict sensitive data access based on user roles (e.g., admin, user) within the Node.js backend. Add MFA for critical actions, such as logging in to administrative accounts. This ensures that even if passwords are compromised, unauthorized access is still prevented.

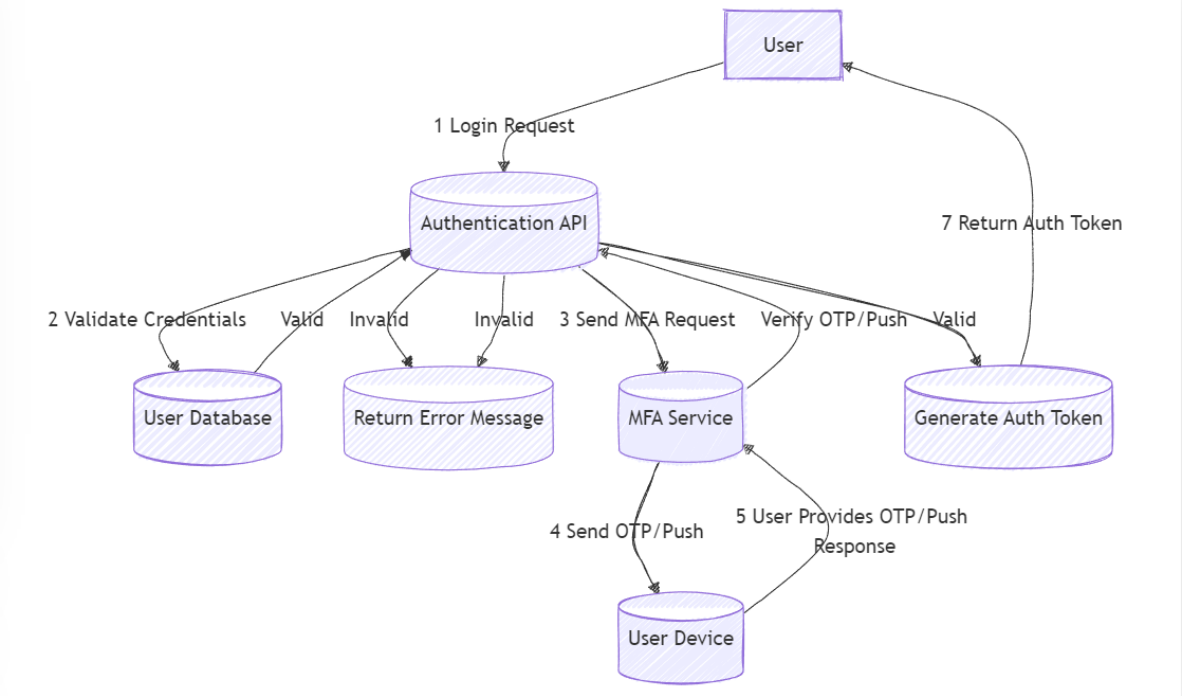
Example: Fitbit Health Solutions uses RBAC and MFA to ensure that health data is only accessible to authorized users.

Node.js Example: For RBAC, you can implement roles in your database and control API access through middleware functions. Libraries like passport.js and speakeasy for TOTP-based MFA can handle user authentication securely.

Code Snippet eg:

```
function ensureAdmin(req, res, next) {
  if (req.user && req.user.role === 'admin') {
    next();
  } else {
    res.status(403).send('Forbidden');
  }
}
```

Basic MFA Architecture



3. Data Sharing and Consent Management:

Implementation: Ensure that users give explicit consent before collecting or sharing any data, especially location or activity data. Implement a consent management system where users can view and control what data is shared, for how long, and with whom. This is particularly important for human activity detection apps, as users' movement patterns are sensitive.

Example: Google Fit and Apple HealthKit both require user consent before sharing data with third-party applications. Users can revoke access at any time.

Node.js Example: Use an API endpoint to manage user consent with fields in your database that track what permissions a user has granted

4. User Rights and Data Management (Access, Correction, and Deletion):

Implementation: Create API endpoints for users to access, update, and delete their data from the system. Use JWT tokens for secure API calls, ensuring that only the rightful owner can modify their data.

Example: Strava, a fitness tracking app, allows users to download, delete, or correct their data. Strava also allows users to opt out of specific data tracking features.

Node.js Example: Build secure API routes to handle these requests, following best practices like logging and encrypting sensitive data deletions.

Code Snippet Eg:

```
app.delete('/user/:id', (req, res) => {
  db.deleteUser(req.params.id).then(() => res.status(200).send('User data deleted'));
});
```

5. Security Audits and Incident Response:

Implementation: Conduct regular security audits of both the Node.js backend and Android client for vulnerabilities. Maintain an Incident Response Plan that can quickly react to potential breaches, including notifying affected users and authorities.

Example: After the 2022 data breach, Medibank significantly enhanced their IT security and incident response plans to prevent future breaches

Node.js Implementation: Integrate security scanning tools like Snyk or npm audit into your CI/CD pipeline to detect vulnerabilities. Also, create an incident response API route to log security incidents and notify users via emails or push notifications.

Code Snippets Eg:

```
app.post('/incident-report', (req, res) => {
  // Log incident details and notify users
  logIncident(req.body.details);
  notifyAffectedUsers();
  res.status(200).send('Incident reported');
});
```

Implementation of Security on Guardian Monitor App

Feature	Description	Status
Authentication	JWT tokens are used to secure all API endpoints the frontend will interact with.	Completed
Data Encryption	API is hosted on an HTTPS server to ensure encrypted data transmission.	Completed
Role-based Access	Roles are implemented for different user types to control access to data based on user permissions.	Completed
MFA (Multi-Factor Authentication)	Integrating an SMTP server to send a code to registered email addresses during login as part of MFA.	Pending

References:

1] Australian Cyber Security Centre (ACSC) Guidelines, Available: https://www.cyber.gov.au/

2] Therapeutic Goods Act (TG Act), Available: https://www.tga.gov.au/

3] Privacy Act 1988 (Australia), Available: https://www.oaic.gov.au/

4] Healthcare Identifiers Act 2010, Available: https://www.digitalhealth.gov.au/

5] Notifiable Data Breaches (NDB) Scheme, Available: https://www.oaic.gov.au/privacy/notifiable-data-breaches

6] He, D., Naveed, M., Gunter, C. A., & Nahrstedt, K. (2014). Security Concerns in Android mHealth Apps. AMIA ... Annual Symposium proceedings. AMIA Symposium, 2014, 645–654. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4419898/

7] Medibank breach: Security failures revealed (lack of MFA among them) https://www.helpnetsecurity.com/2024/06/18/medibank-breach-security-failures/

8] 11 Best Practices to Secure your Nodejs API, Available: https://www.indusface.com/blog/how-to-secure-nodejs-api/

9] Security Best Practices, Available: https://nodejs.org/en/learn/getting-started/security-best-practices