1. Project Overview

Melbourne has become more dependent on car travel since COVID compared with similar cities across the world. A major source of congestion in Melbourne's CBD is the time drivers spend searching for a parking space. With a high rate of car ownership, this search can be difficult on average, drivers spend 17 hours a year searching for a parking spot within Melbourne city. This forces Melbourne commuters to continually drive around the city until one is free, increasing congestion, pollution and wasting time. The root cause of this frustration and inefficiency is the absence of real-time data on available parking spaces, which leaves drivers in frustration.

How might we provide drivers with real-time parking information to reduce search times and ease congestion?

The following system diagram will provide an insight for the system, showing how each component is set/interacted：

图示

AI 生成的内容可能不正确。

System Components:

User Interaction Layer: Users engage with a web interface that is intuitive and informational, without the system recording any personal data.

External Data Sources: Periodic imports of updated CSV files ensure the platform's data remains current.

GitHub Repository: Hosts the application code for version control and facilitates secure, consistent updates and deployment.

Security Protocols: Emphasises secure data transmission and server configuration, with SSL encryption and regular security audits.

By aligning each system component with the platform's evolving features and the target audience's needs, this overview provides a clear picture of the project.

2. System Security Overview

For any Internet application, ensuring security is an integral part. Security is not only related to the data security and information property security of web users and operators but also ensures the availability of network applications. Through research and discussion among team members, the team was able to identify potential vulnerabilities that may exist in the project. A table for vulnerabilities and remediation will be shown as below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Vulnerability** | **Risk Rating** | **Remediation Steps** |
| 1 | Identification & Authentication Failures |  | MFA, session management, account lockout |
| 2 | SQL Injection |  | Strict input validation |
| 3 | Broken Access Control |  | Role-based access, privilege control |
| 4 | Security Misconfiguration |  | Hardened configs, regular patching |
| 5 | Insecure Cryptographic Storage |  | AES-256 encryption, hashed passwords |
| 6 | Cross-Site Scripting (XSS) |  | Input sanitization, output escaping |
| 7 | Improper Input Validation |  | Backend schema validation |
| 8 | Insufficient Logging & Monitoring |  | Centralized logs, real-time anomaly alerts |
| 9 | Insecure APIs |  | Secure API endpoints with authentication, rate limiting, and input validation. |

3. Security Awareness

Due to the short project development cycle and large project size, the team needs to ensure that they have considerable security awareness in each development life cycle and operation. This safety awareness are listed below：

1. Access control
2. Data protection
3. Threat detection and response
4. Network security
5. Software security
6. Security trains

4. Ethical, Legal, Security and Privacy Issues

a. Ethical Considerations

b. Legal Compliance

c. Privacy Policy

Maintaining a focus on these ethical, legal, security, and privacy considerations is critical for the trustworthiness and sustainability of the platform. The project team promises to use user data reasonably within the scope permitted by relevant ethics and regulations and ensure that user privacy data is not abused or used without permission. Regular reviews and updates to these policies should be conducted to keep pace with evolving ethical standards and regulatory requirements.

5. Risk Analysis

The following table provides an overview of the potential risks associated with each identified vulnerability, considering how likely the risk is to occur (Likelihood), the level of impact it would have on the system if it were to occur (Impact), and the overall Risk Rating which is often a product of Likelihood and Impact. The Remediation Strategies column outlines steps to mitigate each risk.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Risk** | **Vulnerability** | **Likelihood** | **Impact** | **Risk Rating** | **Remediation Strategies** |
| 1 |  | Identification & Authentication Failures |  |  |  | MFA, session management, account lockout |
| 2 |  | SQL Injection |  |  |  | Strict input validation |
| 3 |  | Broken Access Control |  |  |  | Role-based access, privilege control |
| 4 |  | Security Misconfiguration |  |  |  | Hardened configs, regular patching |
| 5 |  | Insecure Cryptographic Storage |  |  |  | AES-256 encryption, hashed passwords |
| 6 |  | Cross-Site Scripting (XSS) |  |  |  | Input sanitization, output escaping |
| 7 |  | Improper Input Validation |  |  |  | Backend schema validation |
| 8 |  | Insufficient Logging & Monitoring |  |  |  | Centralized logs, real-time anomaly alerts |
| 9 |  | Insecure APIs |  |  |  | Secure API endpoints with authentication, rate limiting, and input validation. |

6. Risk Mitigation Strategies

* System Protection
  + Input Sanitise
  + Secure API Access
  + Enhanced Authentication
  + Network Security Configurations
* Data Protection
  + Encryption Protocols
  + Data Masking
* General Measures
  + Regular Security Audits
  + Backup

By implementing these risk mitigation strategies, the platform can significantly enhance its system and data protection capabilities, leading to a more robust security posture that adapts to evolving threats in the digital landscape.