README.md

```
 ______   ________   ______   _________   ______
/_____/\ /_______/\ /_____/\ /________/\ /_____/\
\:::_ \ \\::: _  \ \\:::_ \ \\__.::.__\/ \:::_:\ \
 \:(_) \ \\::(_)  \ \\:(_) ) )_ \::\ \      /_\:\ \
  \: ___\/ \:: __  \ \\: __ `\ \ \::\ \     \::_:\ \
   \ \ \    \:.\ \  \ \\ \ `\ \ \ \::\ \    /___\:\ '
    \_\/     \__\/\__\/ \_\/ \_\/  \__\/    \______/
```

System Security

Plan for make our system secure

Infrastructure and data security is of utmost importance to our exciting startup revolutionizing solar panel installation through a mobile app and web page. Our commitment to protecting customer information and the secure operation of our services is essential to maintaining the trust of our users. In this security plan we address key protection aspects following the guidelines provided by the OWASP Top 10 for 2021.

OWASP Top 10 - Plan

ID	Category	Importance to our system
1	Injection Attacks	We have to ensure that all user inputs are properly validated and sanitized to prevent injection attacks (HTML injection). This is critical our Python backend when interacting with the MySQL database. For this, we can use prepared statements or ORM libraries to mitigate SQL injection risks. We also need to ensure that all queries to the MySQL database are parameterized to prevent SQL injection.
2	Broken Authentication	It is fundamental to implement strong authentication mechanisms for both your mobile app and web frontend. Use secure protocols and ensure passwords are securely stored and hashed in the MySQL database. Implement account lockout and password reset mechanisms to protect against unauthorized access.
3	Sensitive Data Exposure	It is essential to implement strong authentication mechanisms for both the mobile application and the web frontend. Use secure protocols and ensure passwords are securely stored and encrypted in the MySQL database. Implementation of account lockout and password reset mechanisms to protect against unauthorized access.
4	XML External Entities (XXE)	Protect the system against XML-based attacks by disabling external entity processing and validating XML inputs. We have to make sure XML parsing libraries are securely configured in the Python backend. To do this, we can implement role-based access control (RBAC) to ensure that employees can only access the data and functions they need for their work.

5	Broken Access Control	Protect the system against XML-based attacks by disabling external entity processing and validating XML inputs. We have to make sure XML parsing libraries are securely configured in the Python backend. To do this, we can implement role-based access control (RBAC) to ensure that employees can only access the data and functions they need for their work.
6	Security Misconfigurations	We must periodically audit and evaluate the Kubernetes cluster and AWS environment to identify and rectify any security misconfigurations. To do this we can use AWS Identity and Access Management (IAM) roles and Kubernetes security best practices.
7	Cross-Site Scripting (XSS)	User-generated content and data must be sanitized and validated before being presented in the web interface. To do this, we can use security libraries and frameworks to mitigate XSS vulnerabilities in Next.js and JavaScript applications.
8	Insecure Deserialization	We must be careful about deserialization vulnerabilities, especially when processing data from untrusted sources. We must implement input validation and reliable serialization libraries. This starts from our backend developed with Python.
9	Using Components with Known Vulnerabilities	We should periodically monitor security advisories, apply patches, or, a much better option, quickly update vulnerable components. In our case our dependencies and the JavaScript and Python libraries.
10	Insufficient Logging and Monitoring	We must do comprehensive recording and monitoring throughout the system. To do this, we can configure alerts for unusual or suspicious activities and periodically review logs to detect and respond to possible security incidents.

Conclusion

In conclusion, our security plan, guided by the OWASP Top 10 for 2021, prioritizes safeguarding customer data and maintaining system integrity. Through measures like input validation, strong authentication, encryption, and access control, we're committed to ensuring a secure environment for our solar panel installation app. Our proactive approach, incident response planning, and continuous vigilance underscore our dedication to security and customer trust.