**Question List and Prompt Strategy**

|  |  |  |  |
| --- | --- | --- | --- |
| Question | Prompt strategy | Examples | GPT Improved Answer |
| Name of application |  | Teacher Helper – Azure OpenAI LLM | Teacher Helper – Azure OpenAI LLM |
| Introduction of the business functions and general aspects |  |  |  |
| Contact person |  |  | John Doe |
| Describe the main business purposes of the application. Companies or consumers in scope | Improve writing and translate into target language | Cloud Based AI Application focused on Teachers being able to grade and rate their students exams. Exams are scanned via scanner and is in PDF Format. The teacher will have a dashboard where PDF Files canbe uploaded. Steps include\_  1. OCR Function to change the handwritted words to machine readable words. 2. Administrative Dashboard to configure the commands: how to rate every exam question.  E.g : history will require more prompt according to its tasks / knowledge base  All informations will be managed by the teacher/backend support staff: Click on button and the exam will be rated/ graded automatically. Then, grades will be received for each exam and total grade for the entire exams.   The teacher is able to edit and confirm/modify the grade.  The system must be capable to store the results and in versions.  The system will be used for 300 pupils and 10 teachers of Middle School. | This Cloud-Based AI Application is designed to assist teachers in grading and rating student exams efficiently. Exams, scanned and saved as PDF files, can be uploaded through a teacher's dashboard. Key features include:  1. OCR Functionality: Converts handwritten text to machine-readable words.  2. Administrative Dashboard: Allows configuration of grading criteria for each exam question, tailored to subjects like history which may require more detailed prompts based on task requirements and knowledge base.  Teachers and backend support staff can manage all information. With a simple click, exams are automatically graded, providing individual and cumulative grades for each student. Teachers can review, edit, and confirm or modify grades as needed.  The system can store results and maintain version histories, accommodating up to 300 students and 10 middle school teachers. |
| Describe approximate number of users and where are they located, countries and region | Depending on regions there must be specific requirements. E.g. in EU follow GDPR, data should not be hosted outside of EU. In other countries less restrictions. In China and Russia data must be kept inside the same country. Add this information into the prompt and let GPT generate the controls. | Germany – used for schools. 300 -pupils + 20-teachers + 5 admin users.  300 pupils, but 20 teachers will be the only one who will have access to the application | For Germany, the application will be used in schools with 300 pupils, 20 teachers, and 5 admin users. Only the 20 teachers will have access to the application. As per GDPR requirements, data must be hosted within the EU, and all data processing must comply with GDPR standards. |
| Describe the level of confidentiality, integrity and availability. Examples for confidentiality: public, internal, confidential, secret. Example for availabilty: standard or critical. Example for integrity: standard or critical. If the app includes personal sensitive data such as credit card number, motion data, sexual preference etc.? All can be can also be implemented as check boxes. | Extract the information and use them across the entire prompting. E.g. this is a confidential application (public, internal, confidential, secret). Based on this classification recommend the corresponding controls.  Personal sensitive data -> strict controls | Should be confidential due to the grades of every pupil. Confidentiality not only related to grades and personal info but also the history of the grades and exam questions. Latest exams and questions will be added to the history/system. **Integrity** important topic specially in props and should not be modified. Can face risk that it can be affected in malicious way and impact the results of pupil’s exams. **Availability** is not very critical. It does include personal sensitive data. Personal data, but is not personal-sensitive data. (Name will be written etc.) | This is a confidential application due to the inclusion of pupils' grades, personal information, and exam questions. The confidentiality of the system must be maintained to protect the history of grades and exam questions, including the latest additions. Integrity is crucial, particularly for exam props, and should remain unaltered to prevent malicious impacts on pupils' exam results. While availability is not highly critical, the application contains sensitive personal data such as names, which must be protected. Based on this classification, the corresponding controls should include robust encryption, access controls, regular integrity checks, and secure storage solutions to ensure confidentiality and integrity. |
| Kind of application: SaaS, completely Self developed, or individual coding based on existing products |  | Self Developed application based on requirements of the Teachers from the given organization(School). It is individual coding – solution is made from scratch. | Self-developed application based on the requirements of the teachers from the given school organization. It is an individual coding project created from scratch. |
| Industry of the application (Example: automotive, health, public sector, financial, consumer, telecom, machinery, chemistry, service, consulting, education) | Ask if any industry specific security requirements are necessary. | Education | Are there any specific security requirements necessary for the education industry? |
| Any certification available (e.g. SOC2 type2, ISO27001 etc.). | In case of SaaS the certification must be related to the SaaS software. In other cases the certification is related to the (cloud) hosting environment. Judge if the certification is available or not, and if it is sufficient. | N/A | In the case of SaaS, the certification must pertain to the SaaS software. For other scenarios, the certification should be associated with the (cloud) hosting environment. Assess if the certification is available and sufficient. |

**Authentication/authorization**

|  |  |  |  |
| --- | --- | --- | --- |
| Question | Prompt strategy |  |  |
| Do you use a company wide SSO system or a built-in authentication of the application? | Say that company authorized SSO should always be preferred over the built-in authentication of the application. Only in the case of few users, it makes no sense to integrate SSO since it causes too much efforts. | No- since it is a small solution, system will use its built-in authentication. User gets account authentication through email address. | Since this is a small-scale solution, we will use the system's built-in authentication. Users will receive account authentication via their email addresses. |
| What authentication mechanism is being used for the web application? Is it based on passwords, multi-factor authentication, or some other method? | Describe the company’s password policy, MFA etc. | Password based authentication. Password consists of 10 Characters which is enforced in the application and consist of Capital Letters, Number and Symbols. Password must be rotated every 3 months. The rotation is imposed by the system.  Currently, for assessing passwords there is no protection(Like account suspension policy or a captcha) | Our company employs a robust password-based authentication policy. Passwords must be at least 10 characters long and include a combination of capital letters, numbers, and symbols. Passwords must be updated every 3 months, with the system enforcing this rotation. Currently, there are no additional protections in place, such as account suspension policies or CAPTCHA, to assess password security. |
| Is account creation and management secure? | Describe a secure account creation and management process. Describe how to prevent unauthorized account creation or modification. | Yes. Account creation is a self registration process but a Principal must approve the user and assign user related role. | Account creation involves a self-registration process, but a Principal must approve the user and assign the appropriate role. This ensures secure account creation and management, preventing unauthorized account creation or modification. |
| What authorization mechanism is being used for the web application? Describe technology | Based on the described technology, describe best practice according | Role-based access, meaning there are different user roles: admin, principal, teacher assigned to a subject. User Role is managed by the application itself by using cookie- in the cookie you will have a token and includes publishing of the user and this token will be validated in every request of the application. | Role-based access control (RBAC) is implemented to manage different user roles such as admin, principal, and teacher assigned to a subject. The application itself manages these user roles using cookies. Each cookie contains a token that identifies the user and their role. This token is validated with every request to ensure proper authorization and access control. |
| Where are user credentials stored? Which programming language or framework is used. | Describe which function should be used based on the given programming language and framework. Mention hash and salt. | User credentials are stored in database in a hashed and salted form. As a programming function we use Bcrypt | User credentials are stored in the database in a hashed and salted form using the Bcrypt function. |
| Is there any sort of rate limiting or account lockout policies in place to prevent brute force attacks? Name any products used if available | Describe the right account lockout policies and rate limiting mechanism based on the product, if any. | Currently no, for assessing passwords there is no protection(Like account suspension policy or a captcha) | At present, there are no protective measures in place for password assessment, such as account suspension policies or CAPTCHA mechanisms. |
| Does the system implement secure session management? Which product or technology is used? How are session tokens handled and protected? Name the technology | Describe how should session tokens be handled and protected based on the product or technology. Describe what is the best practice for secure session management. | Role-based access, meaning there are different user roles: admin, principal, teacher assigned to a subject. User Role is managed by the application itself by using cookie- in the cookie you will have a token and includes publishing of the user and this token will be validated in every request of the application. | Session tokens should be handled with utmost care to ensure secure session management, especially in an application with role-based access for different user roles such as admin, principal, and teacher. The application should manage user roles using cookies that contain a token representing the user's role. This token should be securely validated with every request to the application.  To protect these session tokens, follow these best practices:  1. \*\*Use Secure Cookies:\*\* Mark cookies as secure to ensure they are only transmitted over HTTPS.  2. \*\*HttpOnly Flag:\*\* Set the HttpOnly attribute on cookies to prevent access to the token via client-side scripts.  3. \*\*SameSite Attribute:\*\* Use the SameSite attribute to restrict how cookies are sent with cross-site requests, reducing the risk of Cross-Site Request Forgery (CSRF) attacks.  4. \*\*Token Expiry:\*\* Implement token expiration to limit the duration of a session, forcing users to re-authenticate periodically.  5. \*\*Regenerate Tokens:\*\* Regenerate session tokens after login and periodically during a session to minimize the risk of session hijacking.  6. \*\*Store Minimal Information:\*\* Keep the cookie content minimal and do not store sensitive user information directly in the cookie.  7. \*\*Encryption:\*\* Encrypt the token to add an additional layer of security.  8. \*\*Validate Tokens:\*\* Ensure tokens are validated on the server side for each request to verify the user's role and permissions.  By adhering to these practices, you can significantly enhance the security of session management in your application. |
| How does the application manage user roles and permissions? Name all planned user roles. | Describe the user roles and their permissions in best practice. | Four User Roles, and user groups for different subjects.  Every non-admin/principal user can only see the documents of their class/subject.  Admin and principal can see all.  Admin can do administrative/it tasks.  Principal has access to all exams and grades but only has read only access and approval. Can also draw reports to see the overview for each classes. | There are four user roles with specific permissions:  1. \*\*Admin\*\*: Responsible for administrative and IT tasks. Has access to all documents and data across all classes and subjects.  2. \*\*Principal\*\*: Has read-only access to all exams and grades, and can approve them. Can generate reports to get an overview of each class.  3. \*\*Teachers\*\*: Can only view and manage documents related to their own class or subject.  4. \*\*Students\*\*: Can only view documents related to their own class or subject.  Additionally, users are grouped by different subjects to streamline access and management. |
| Does the application have a secure password reset or account recovery mechanism? | Describe how a secure password reset or account recovery looks like. | Yes, there is a set password reset function where the users can click it and reset password. Resetting password: directly change the password and click button to confirm. | A secure password reset or account recovery process typically involves the following steps:  1. User initiates the password reset by clicking on a "Forgot Password" link.  2. User is prompted to enter their registered email address or username.  3. A verification code or password reset link is sent to the user's registered email address or phone number.  4. User clicks on the link or enters the verification code to access the password reset page.  5. User is prompted to answer security questions or complete additional verification steps, such as two-factor authentication, to confirm their identity.  6. Once verified, the user is allowed to create a new password.  7. User enters the new password and confirms it by entering it again.  8. User clicks a button to save the new password.  9. A confirmation message is sent to the user's registered email address or phone number, informing them of the password change.  10. The user can now log in with the new password. |
| Does the web application use micro services and name the products used for the micro services. | Describe the proper authentication this micro service. | Yes, web application consist of multiple micro services:   1. User administration 2. Authentication 3. OCR (recognition of handwritten letters) 4. Automatic rating with the help of Azure OpenAI 5. Managing the prompts 6. Admin   All of this microservices are stored in a cluster in every container, so 1 container and 1 microservice is stored in a Kubernetes cluster. | The web application comprises several microservices, including:  1. User Administration  2. Authentication  3. OCR (Optical Character Recognition for handwritten letters)  4. Automatic Rating using Azure OpenAI  5. Prompt Management  6. Admin  Each of these microservices is deployed in a separate container within a Kubernetes cluster. |

Application Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| Question | Prompt strategy | Sample User Answer | Sample ChatGPT Answer |
| Describe the main technology used for the application. Main framework, programming language, functions. | Improve the writing and translate into target language | Python Django as Mid Layer, ReactJS as frontend, Azure SQL DB (MySQL , Postgres, etc), Azure Blob Storage (to store the scanned files) | Python Django as the middle layer, ReactJS for the frontend, Azure SQL Database (MySQL, Postgres, etc.), and Azure Blob Storage for storing scanned files |
| Describe the technology of the frontend and used products | Figure out any known security weakness of the frontend technology and framework and describe the countermeasures. Consider also OWASP top 10 for mobile app. | Frontend uses Javascript – single page and will be called by the user. It is kind of a dashboard wherein the user is the teacher and the dashboard has login/logout button and the teacher can upload materials. | The frontend utilizes JavaScript to create a single-page application that serves as a dashboard for teachers. This dashboard includes login and logout functionality, allowing teachers to upload materials easily. |
| Describe the technology stack of the middle layer and used products | Figure out any known security weakness of the middleware technology and framework and describe the countermeasures. Consider also OWASP top 10. | For the midlayer, we are using Apache Web Server and on this Web Server we will deploy Python Django web application. Python Django application is serving all request from frontend but is still responsible for all the workflow. It will call different microservices and will also call external services (Azure OpenAI, OCR Service). Middle Layer (Django Application) will also access the data base and the blob storage. | For the midlayer, we are using Apache Web Server to deploy a Python Django web application. The Django application handles all requests from the frontend and manages the workflow. It interacts with various microservices and external services such as Azure OpenAI and OCR Service. Additionally, the Django application accesses the database and blob storage. |
| Describe the backend layer or used database | Figure out any known security weakness of the backend technology and framework and database, and describe the countermeasures. Consider also OWASP top 10. | We are using platform as database service from Azure. The Database is used to store the user administration information, status information, prompt, and temporary process. | To ensure the security of our Azure Database service, which stores user administration information, status information, prompts, and temporary processes, we should consider the following countermeasures:  1. \*\*SQL Injection Prevention:\*\* Use parameterized queries and stored procedures to prevent SQL injection attacks. Regularly update and patch database management systems.  2. \*\*Access Control:\*\* Implement strict access controls using Azure's role-based access control (RBAC). Ensure only authorized personnel have access to the database.  3. \*\*Encryption:\*\* Utilize Azure's built-in encryption at rest and in transit. Ensure data is encrypted using strong encryption standards like AES-256.  4. \*\*Audit Logging:\*\* Enable auditing to log access and changes to the database. Use Azure Monitor and Azure Security Center to track and respond to suspicious activities.  5. \*\*Firewall Configuration:\*\* Configure Azure SQL Database firewalls to limit access to trusted IP addresses only.  6. \*\*Regular Backups:\*\* Perform regular automated backups and ensure they are stored securely. Test backup restoration procedures periodically.  7. \*\*Input Validation:\*\* Validate all input data on the server side to prevent XSS, CSRF, and other injection attacks.  8. \*\*Compliance:\*\* Ensure compliance with relevant data protection regulations and standards such as GDPR, HIPAA, and SOC 2 by configuring and managing Azure Policy and Blueprints.  9. \*\*Security Updates:\*\* Regularly apply security patches and updates to the database service to mitigate known vulnerabilities.  10. \*\*OWASP Top 10:\*\* Regularly review and address the OWASP Top 10 security risks, ensuring that the application and database are safeguarded against common vulnerabilities such as broken authentication, sensitive data exposure, and security misconfigurations. |
| Describe the relation between frontend, middle layer or backend. If any layer is not available, just skip. | Judge if it is a good architecture design. | It is a 3 tier architecture that is most likely, static, middle and backend layer (data layer). There is an isolation between different layers.   * Can also skip | It is a 3-tier architecture consisting of the static layer, middle layer, and backend layer (data layer), with isolation between the different layers. |
| Describe any library, dependencies used | Improve the writing and translate into target language. | Library for authentication, security validation, library to manage the ai services | Biblioteca para autenticação, validação de segurança e gerenciamento de serviços de IA |
|  |  |  |  |
|  |  |  |  |

Cloud Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| Question | Prompt strategy |  |  |
| Name of the cloud provider, e.g. Azure, AWS, GCP |  | Azure | - |
| Name all cloud resources used for this application such as VM, App service, container service, security groups, firewall, private link, database, application gateway, forward proxy, load balancer, security proxy, data storage, | Describe the hardening measures for every of this cloud resource depending on the classification. | * Azure Functions * Application Gateway * Firewall * Database * Blob Storage * Azure Kubernetes Service (because of multiple micro services) * Azure OpenAI Service * Azure AI Computer Vision (for OCR) | - Azure Functions: Implement network restrictions, use managed identities, enable authentication and authorization, enforce HTTPS, and apply role-based access control (RBAC).  - Application Gateway: Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), and enable diagnostic logging.  - Firewall: Configure application rules and network rules, enable threat intelligence-based filtering, implement logging and monitoring, and apply strict access control policies.  - Database: Enable encryption at rest and in transit, use automatic backups, implement Advanced Threat Protection, enforce strong authentication and access control, and regularly update and patch.  - Blob Storage: Enable encryption at rest, use private endpoints, enforce access policies and Shared Access Signatures (SAS), enable logging and monitoring, and configure data retention policies.  - Azure Kubernetes Service: Implement network policies, use managed identities, enable RBAC, configure pod security policies, regularly update and patch the cluster, and use Azure Policy for enforcement.  - Azure OpenAI Service: Enable network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, and ensure data encryption in transit and at rest.  - Azure AI Computer Vision: Restrict access using NSGs, enable authentication and authorization, monitor and log access, enforce data encryption in transit, and apply RBAC for resource access. |
| Describe the relation between all the cloud resources | Judge if this is a good cloud security architecture design. Recommend security best practices for this cloud settings. |  | The relationship between these components and recommended security best practices are as follows:  - Azure Functions: Implement network restrictions, use managed identities, enable authentication and authorization, enforce HTTPS, and apply role-based access control (RBAC).  - Application Gateway: Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), and enable diagnostic logging.  - Firewall: Configure application rules and network rules, enable threat intelligence-based filtering, implement logging and monitoring, and apply strict access control policies.  - Database: Enable encryption at rest and in transit, use automatic backups, implement Advanced Threat Protection, enforce strong authentication and access control, and regularly update and patch.  - Blob Storage: Enable encryption at rest, use private endpoints, enforce access policies and Shared Access Signatures (SAS), enable logging and monitoring, and configure data retention policies.  - Azure Kubernetes Service: Implement network policies, use managed identities, enable RBAC, configure pod security policies, regularly update and patch the cluster, and use Azure Policy for enforcement.  - Azure OpenAI Service: Enable network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, and ensure data encryption in transit and at rest.  - Azure AI Computer Vision: Restrict access using NSGs, enable authentication and authorization, monitor and log access, enforce data encryption in transit, and apply RBAC for resource access. |
| Describe if there are any connections between the cloud resources and company’s onprem environment | Judge if the connection is secure considering the classification. What is the preferred architecture? | No – pure cloud environment without further connection to the school’s own environment | A pure cloud environment without any connection to the school’s own environment is secure. The preferred architecture would be a fully cloud-based solution. |
| Describe how the user can access the web application, from the internet, from company’s intranet. | Judge if the access is secure regarding the classification. | The users can access it from the internet directly. | Access from the internet is direct. Evaluate if it meets the security classification. |
| Describe monitoring and evaluation services | Judge if that is sufficient and mention that they should be connected to company’s central monitoring and SIEM solutions. | No such monitoring solutions provided | The provided monitoring solutions are insufficient and should be integrated with the company’s central monitoring and SIEM solutions. |
| Describe the current cloud governance model | Judge if the cloud governance model is sufficient and recommend improvements. | Basic cloud governance managed by the school’s IT Department | The current cloud governance model, managed by the school’s IT Department, is a good starting point. However, to ensure it is sufficient, consider enhancing it by implementing regular audits, establishing clear policies and procedures, incorporating automated compliance checks, and providing ongoing training for staff. |
|  |  |  |  |

Final prompting:

Create a report with following structure:

Management summary and mention the business function briefly and the most critical controls.

Introduction of the business functions and general aspect description.

Describe authentication/authorization concept.

Describe the application architecture and its recommended controls.

Describe controls for cloud architecture.

Let general information such as application name, confidentiality etc. always be integrated in the following chapters.

A further chapter for visualization of the architecture (backlog).

**Notes and Evaluation:**

* This is the prompt that was used :

{

                    "role": "user",

                    "content": f"""

                    Improve the text that is provided and return the final text. For reference here is the information

                    question: {question}

                    prompt strategy: {prompt},

                    user input: {data}

                    Return only the answer and no explanation for the user is required.

                    Do not return follow up questions.

                    The text format does not include bold/itallic/underline.

                    """,

                }

* **Prompt Strategy Improvement:** The elements highlighted in gray indicate areas where the prompt stratefy requires enhancement.
* **Language Capabilities:** The feature for converting to language capabilities has been temporarily disabled until a language selection option is implemented.
* **Judging Keyword:** The use of the keyword “Judge” and “Ask” has proven ineffective in generating accurate prompt. It is recommended that prompt be structured to reflect and objective format for better clarity and effectiveness.
* Should improve the prompt to remove the \*\* for bold generated texts given by Azure OpenAI

# Collection of GPT Answers

## General Information:

|  |
| --- |
| Teacher Helper – Azure OpenAI LLM |
|  |
| John Doe |
| This Cloud-Based AI Application is designed to assist teachers in grading and rating student exams efficiently. Exams, scanned and saved as PDF files, can be uploaded through a teacher's dashboard. Key features include:  1. OCR Functionality: Converts handwritten text to machine-readable words.  2. Administrative Dashboard: Allows configuration of grading criteria for each exam question, tailored to subjects like history which may require more detailed prompts based on task requirements and knowledge base.  Teachers and backend support staff can manage all information. With a simple click, exams are automatically graded, providing individual and cumulative grades for each student. Teachers can review, edit, and confirm or modify grades as needed.  The system can store results and maintain version histories, accommodating up to 300 students and 10 middle school teachers. |
| For Germany, the application will be used in schools with 300 pupils, 20 teachers, and 5 admin users. Only the 20 teachers will have access to the application. As per GDPR requirements, data must be hosted within the EU, and all data processing must comply with GDPR standards. |
| This is a confidential application due to the inclusion of pupils' grades, personal information, and exam questions. The confidentiality of the system must be maintained to protect the history of grades and exam questions, including the latest additions. Integrity is crucial, particularly for exam props, and should remain unaltered to prevent malicious impacts on pupils' exam results. While availability is not highly critical, the application contains sensitive personal data such as names, which must be protected. Based on this classification, the corresponding controls should include robust encryption, access controls, regular integrity checks, and secure storage solutions to ensure confidentiality and integrity. |
| Self-developed application based on the requirements of the teachers from the given school organization. It is an individual coding project created from scratch. |
| Are there any specific security requirements necessary for the education industry? |
| In the case of SaaS, the certification must pertain to the SaaS software. For other scenarios, the certification should be associated with the (cloud) hosting environment. Assess if the certification is available and sufficient. |

## Authentication Authorization

|  |
| --- |
|  |
| Since this is a small-scale solution, we will use the system's built-in authentication. Users will receive account authentication via their email addresses. |
| Our company employs a robust password-based authentication policy. Passwords must be at least 10 characters long and include a combination of capital letters, numbers, and symbols. Passwords must be updated every 3 months, with the system enforcing this rotation. Currently, there are no additional protections in place, such as account suspension policies or CAPTCHA, to assess password security. |
| Account creation involves a self-registration process, but a Principal must approve the user and assign the appropriate role. This ensures secure account creation and management, preventing unauthorized account creation or modification. |
| Role-based access control (RBAC) is implemented to manage different user roles such as admin, principal, and teacher assigned to a subject. The application itself manages these user roles using cookies. Each cookie contains a token that identifies the user and their role. This token is validated with every request to ensure proper authorization and access control. |
| User credentials are stored in the database in a hashed and salted form using the Bcrypt function. |
| At present, there are no protective measures in place for password assessment, such as account suspension policies or CAPTCHA mechanisms. |
| Session tokens should be handled with utmost care to ensure secure session management, especially in an application with role-based access for different user roles such as admin, principal, and teacher. The application should manage user roles using cookies that contain a token representing the user's role. This token should be securely validated with every request to the application.  To protect these session tokens, follow these best practices:  1. \*\*Use Secure Cookies:\*\* Mark cookies as secure to ensure they are only transmitted over HTTPS.  2. \*\*HttpOnly Flag:\*\* Set the HttpOnly attribute on cookies to prevent access to the token via client-side scripts.  3. \*\*SameSite Attribute:\*\* Use the SameSite attribute to restrict how cookies are sent with cross-site requests, reducing the risk of Cross-Site Request Forgery (CSRF) attacks.  4. \*\*Token Expiry:\*\* Implement token expiration to limit the duration of a session, forcing users to re-authenticate periodically.  5. \*\*Regenerate Tokens:\*\* Regenerate session tokens after login and periodically during a session to minimize the risk of session hijacking.  6. \*\*Store Minimal Information:\*\* Keep the cookie content minimal and do not store sensitive user information directly in the cookie.  7. \*\*Encryption:\*\* Encrypt the token to add an additional layer of security.  8. \*\*Validate Tokens:\*\* Ensure tokens are validated on the server side for each request to verify the user's role and permissions.  By adhering to these practices, you can significantly enhance the security of session management in your application. |
| There are four user roles with specific permissions:  1. \*\*Admin\*\*: Responsible for administrative and IT tasks. Has access to all documents and data across all classes and subjects.  2. \*\*Principal\*\*: Has read-only access to all exams and grades, and can approve them. Can generate reports to get an overview of each class.  3. \*\*Teachers\*\*: Can only view and manage documents related to their own class or subject.  4. \*\*Students\*\*: Can only view documents related to their own class or subject.  Additionally, users are grouped by different subjects to streamline access and management. |
| A secure password reset or account recovery process typically involves the following steps:  1. User initiates the password reset by clicking on a "Forgot Password" link.  2. User is prompted to enter their registered email address or username.  3. A verification code or password reset link is sent to the user's registered email address or phone number.  4. User clicks on the link or enters the verification code to access the password reset page.  5. User is prompted to answer security questions or complete additional verification steps, such as two-factor authentication, to confirm their identity.  6. Once verified, the user is allowed to create a new password.  7. User enters the new password and confirms it by entering it again.  8. User clicks a button to save the new password.  9. A confirmation message is sent to the user's registered email address or phone number, informing them of the password change.  10. The user can now log in with the new password. |
| The web application comprises several microservices, including:  1. User Administration  2. Authentication  3. OCR (Optical Character Recognition for handwritten letters)  4. Automatic Rating using Azure OpenAI  5. Prompt Management  6. Admin  Each of these microservices is deployed in a separate container within a Kubernetes cluster. |

## Application Architecture

|  |
| --- |
| Python Django as the middle layer, ReactJS for the frontend, Azure SQL Database (MySQL, Postgres, etc.), and Azure Blob Storage for storing scanned files |
| The frontend utilizes JavaScript to create a single-page application that serves as a dashboard for teachers. This dashboard includes login and logout functionality, allowing teachers to upload materials easily. |
| For the midlayer, we are using Apache Web Server to deploy a Python Django web application. The Django application handles all requests from the frontend and manages the workflow. It interacts with various microservices and external services such as Azure OpenAI and OCR Service. Additionally, the Django application accesses the database and blob storage. |
| To ensure the security of our Azure Database service, which stores user administration information, status information, prompts, and temporary processes, we should consider the following countermeasures:  1. \*\*SQL Injection Prevention:\*\* Use parameterized queries and stored procedures to prevent SQL injection attacks. Regularly update and patch database management systems.  2. \*\*Access Control:\*\* Implement strict access controls using Azure's role-based access control (RBAC). Ensure only authorized personnel have access to the database.  3. \*\*Encryption:\*\* Utilize Azure's built-in encryption at rest and in transit. Ensure data is encrypted using strong encryption standards like AES-256.  4. \*\*Audit Logging:\*\* Enable auditing to log access and changes to the database. Use Azure Monitor and Azure Security Center to track and respond to suspicious activities.  5. \*\*Firewall Configuration:\*\* Configure Azure SQL Database firewalls to limit access to trusted IP addresses only.  6. \*\*Regular Backups:\*\* Perform regular automated backups and ensure they are stored securely. Test backup restoration procedures periodically.  7. \*\*Input Validation:\*\* Validate all input data on the server side to prevent XSS, CSRF, and other injection attacks.  8. \*\*Compliance:\*\* Ensure compliance with relevant data protection regulations and standards such as GDPR, HIPAA, and SOC 2 by configuring and managing Azure Policy and Blueprints.  9. \*\*Security Updates:\*\* Regularly apply security patches and updates to the database service to mitigate known vulnerabilities.  10. \*\*OWASP Top 10:\*\* Regularly review and address the OWASP Top 10 security risks, ensuring that the application and database are safeguarded against common vulnerabilities such as broken authentication, sensitive data exposure, and security misconfigurations. |
| It is a 3-tier architecture consisting of the static layer, middle layer, and backend layer (data layer), with isolation between the different layers. |
| Biblioteca para autenticação, validação de segurança e gerenciamento de serviços de IA |

## Cloud Architecture

|  |
| --- |
| - Azure Functions: Implement network restrictions, use managed identities, enable authentication and authorization, enforce HTTPS, and apply role-based access control (RBAC).  - Application Gateway: Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), and enable diagnostic logging.  - Firewall: Configure application rules and network rules, enable threat intelligence-based filtering, implement logging and monitoring, and apply strict access control policies.  - Database: Enable encryption at rest and in transit, use automatic backups, implement Advanced Threat Protection, enforce strong authentication and access control, and regularly update and patch.  - Blob Storage: Enable encryption at rest, use private endpoints, enforce access policies and Shared Access Signatures (SAS), enable logging and monitoring, and configure data retention policies.  - Azure Kubernetes Service: Implement network policies, use managed identities, enable RBAC, configure pod security policies, regularly update and patch the cluster, and use Azure Policy for enforcement.  - Azure OpenAI Service: Enable network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, and ensure data encryption in transit and at rest.  - Azure AI Computer Vision: Restrict access using NSGs, enable authentication and authorization, monitor and log access, enforce data encryption in transit, and apply RBAC for resource access. |
| The relationship between these components and recommended security best practices are as follows:  - Azure Functions: Implement network restrictions, use managed identities, enable authentication and authorization, enforce HTTPS, and apply role-based access control (RBAC).  - Application Gateway: Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), and enable diagnostic logging.  - Firewall: Configure application rules and network rules, enable threat intelligence-based filtering, implement logging and monitoring, and apply strict access control policies.  - Database: Enable encryption at rest and in transit, use automatic backups, implement Advanced Threat Protection, enforce strong authentication and access control, and regularly update and patch.  - Blob Storage: Enable encryption at rest, use private endpoints, enforce access policies and Shared Access Signatures (SAS), enable logging and monitoring, and configure data retention policies.  - Azure Kubernetes Service: Implement network policies, use managed identities, enable RBAC, configure pod security policies, regularly update and patch the cluster, and use Azure Policy for enforcement.  - Azure OpenAI Service: Enable network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, and ensure data encryption in transit and at rest.  - Azure AI Computer Vision: Restrict access using NSGs, enable authentication and authorization, monitor and log access, enforce data encryption in transit, and apply RBAC for resource access. |
| A pure cloud environment without any connection to the school’s own environment is secure. The preferred architecture would be a fully cloud-based solution. |
| Access from the internet is direct. Evaluate if it meets the security classification. |
| The provided monitoring solutions are insufficient and should be integrated with the company’s central monitoring and SIEM solutions. |
| The current cloud governance model, managed by the school’s IT Department, is a good starting point. However, to ensure it is sufficient, consider enhancing it by implementing regular audits, establishing clear policies and procedures, incorporating automated compliance checks, and providing ongoing training for staff. |

# Report Sample -1

|  |
| --- |
| **Teacher Helper – Azure OpenAI LLM**  **General Information:**   * **Application Name:** Teacher Helper * **Confidentiality:** High * **Developer:** John Doe   **Management Summary**  **Business Function:** Teacher Helper assists teachers in grading and rating student exams efficiently. Exams, scanned and saved as PDF files, can be uploaded through a teacher's dashboard. Key features include OCR functionality for converting handwritten text and an administrative dashboard for configuring grading criteria.  **Most Critical Controls:**   * Robust encryption * Access controls * Regular integrity checks * Secure storage solutions   **Introduction**  Teacher Helper is designed to streamline the grading process for teachers by automatically grading exams. The application uses OCR to convert handwritten text and Azure OpenAI for automatic grading. Teachers can review, edit, and confirm grades through an intuitive dashboard. The system can store results and maintain version histories, accommodating schools with up to 300 students and 20 teachers. Data is hosted within the EU to comply with GDPR requirements, ensuring the confidentiality and integrity of sensitive information.  **Authentication/Authorization Concept**  Teacher Helper employs built-in authentication where users authenticate via email. Passwords must be at least 10 characters long, including a mix of capital letters, numbers, and symbols, and are hashed and salted using Bcrypt. Passwords are updated every three months. Role-Based Access Control (RBAC) is used to manage access with roles including admin, principal, teacher, and student. Session tokens are managed securely with practices like secure cookies, HttpOnly flags, and token encryption.  **Application Architecture and Recommended Controls**  Teacher Helper consists of several microservices deployed in a Kubernetes cluster:   1. User Administration 2. Authentication 3. OCR 4. Automatic Rating using Azure OpenAI 5. Prompt Management 6. Admin  The frontend is a ReactJS single-page application, while the middle layer is a Python Django web application hosted on Apache Web Server. The backend includes an Azure SQL Database and Azure Blob Storage.  **Recommended Controls:**  * **SQL Injection Prevention:** Use parameterized queries and stored procedures. * **Access Control:** Implement strict RBAC. * **Encryption:** Use AES-256 for data at rest and TLS for data in transit. * **Audit Logging:** Enable auditing with Azure Monitor and Security Center. * **Firewall Configuration:** Restrict access to trusted IP addresses. * **Regular Backups:** Perform and test automated backups. * **Input Validation:** Validate all input data on the server side. * **Compliance:** Ensure GDPR compliance using Azure Policy and Blueprints. * **Security Updates:** Regularly apply updates and patches.   **Controls for Cloud Architecture**  The cloud architecture utilizes various Azure services:   * **Azure Functions:** Implement network restrictions, use managed identities, enable authentication, enforce HTTPS, apply RBAC. * **Application Gateway:** Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), enable diagnostic logging. * **Firewall:** Configure application and network rules, enable threat intelligence filtering, implement logging and monitoring, apply strict access control. * **Database:** Enable encryption, automatic backups, Advanced Threat Protection, strong authentication. * **Blob Storage:** Enable encryption, private endpoints, access policies, logging, data retention. * **Azure Kubernetes Service:** Implement network policies, use managed identities, enable RBAC, configure pod security policies, regular updates, Azure Policy. * **Azure OpenAI Service:** Ensure network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, ensure data encryption. * **Azure AI Computer Vision:** Restrict access using NSGs, enable authentication, monitor and log access, enforce data encryption, apply RBAC.   **Visualization of the Architecture (Backlog)**    **Components:**   1. **User Administration Microservice:** Manages user accounts and roles. 2. **Authentication Microservice:** Handles user authentication. 3. **OCR Microservice:** Converts scanned handwritten text to Text 4. **Automatic Rating Microservice:** Uses Azure OpenAI to automatically grade exams. 5. **Prompt Management Microservice:** Manages the grading criteria and prompts for different subjects. 6. **Admin Microservice:** Provides administrative functionalities and oversight.   **Architecture Layers:**   1. **Frontend Layer:** ReactJS-based single-page application for teacher's dashboard. 2. **Middle Layer:** Python Django web application hosted on Apache Web Server, managing requests and workflows. 3. **Backend Layer:** Azure SQL Database for data storage and Azure Blob Storage for scanned files.   **Security Best Practices:**   * **Use Secure Cookies:** Ensure cookies are only transmitted over HTTPS. * **HttpOnly Flag:** Prevents access to the token via client-side scripts. * **SameSite Attribute:** Restricts cookies with cross-site requests to prevent CSRF attacks. * **Token Expiry and Regeneration:** Limits session duration and minimizes session hijacking risks. * **Minimal Cookie Content:** Avoid storing sensitive user information directly in cookies. * **Token Encryption and Validation:** Adds an additional security layer and ensures server-side validation for each request.  By adhering to these practices and utilizing the recommended controls, Teacher Helper aims to maintain the confidentiality, integrity, and availability of sensitive data, ensuring a secure and efficient grading process for teachers. |

# Prompt Used

|  |
| --- |
| Create a report based on the given information, you do not need to rewrite what is already writted but focus on the prompt points that was mentioned. The prompt points, put it as header. General information should also be mentioned only once. This should be very concise:  Final prompting:  Create a report with following structure:  Management summary and mention the business function briefly and the most critical controls.  Introduction of the business functions and general aspect description.  Describe authentication/authorization concept.  Describe the application architecture and its recommended controls.  Describe controls for cloud architecture.  Let general information such as application name, confidentiality etc. always be integrated in the following chapters.  A further chapter for visualization of the architecture (backlog).  Information:  General Information:  Teacher Helper – Azure OpenAI LLM  John Doe  This Cloud-Based AI Application is designed to assist teachers in grading and rating student exams efficiently. Exams, scanned and saved as PDF files, can be uploaded through a teacher's dashboard. Key features include:  1. OCR Functionality: Converts handwritten text to machine-readable words.  2. Administrative Dashboard: Allows configuration of grading criteria for each exam question, tailored to subjects like history which may require more detailed prompts based on task requirements and knowledge base.  Teachers and backend support staff can manage all information. With a simple click, exams are automatically graded, providing individual and cumulative grades for each student. Teachers can review, edit, and confirm or modify grades as needed.  The system can store results and maintain version histories, accommodating up to 300 students and 10 middle school teachers.  For Germany, the application will be used in schools with 300 pupils, 20 teachers, and 5 admin users. Only the 20 teachers will have access to the application. As per GDPR requirements, data must be hosted within the EU, and all data processing must comply with GDPR standards.  This is a confidential application due to the inclusion of pupils' grades, personal information, and exam questions. The confidentiality of the system must be maintained to protect the history of grades and exam questions, including the latest additions. Integrity is crucial, particularly for exam props, and should remain unaltered to prevent malicious impacts on pupils' exam results. While availability is not highly critical, the application contains sensitive personal data such as names, which must be protected. Based on this classification, the corresponding controls should include robust encryption, access controls, regular integrity checks, and secure storage solutions to ensure confidentiality and integrity.  Self-developed application based on the requirements of the teachers from the given school organization. It is an individual coding project created from scratch.  Are there any specific security requirements necessary for the education industry?  In the case of SaaS, the certification must pertain to the SaaS software. For other scenarios, the certification should be associated with the (cloud) hosting environment. Assess if the certification is available and sufficient.  Authentication Authorization  Since this is a small-scale solution, we will use the system's built-in authentication. Users will receive account authentication via their email addresses.  Our company employs a robust password-based authentication policy. Passwords must be at least 10 characters long and include a combination of capital letters, numbers, and symbols. Passwords must be updated every 3 months, with the system enforcing this rotation. Currently, there are no additional protections in place, such as account suspension policies or CAPTCHA, to assess password security.  Account creation involves a self-registration process, but a Principal must approve the user and assign the appropriate role. This ensures secure account creation and management, preventing unauthorized account creation or modification.  Role-based access control (RBAC) is implemented to manage different user roles such as admin, principal, and teacher assigned to a subject. The application itself manages these user roles using cookies. Each cookie contains a token that identifies the user and their role. This token is validated with every request to ensure proper authorization and access control.  User credentials are stored in the database in a hashed and salted form using the Bcrypt function.  At present, there are no protective measures in place for password assessment, such as account suspension policies or CAPTCHA mechanisms.  Session tokens should be handled with utmost care to ensure secure session management, especially in an application with role-based access for different user roles such as admin, principal, and teacher. The application should manage user roles using cookies that contain a token representing the user's role. This token should be securely validated with every request to the application.  To protect these session tokens, follow these best practices:  1. \*\*Use Secure Cookies:\*\* Mark cookies as secure to ensure they are only transmitted over HTTPS.  2. \*\*HttpOnly Flag:\*\* Set the HttpOnly attribute on cookies to prevent access to the token via client-side scripts.  3. \*\*SameSite Attribute:\*\* Use the SameSite attribute to restrict how cookies are sent with cross-site requests, reducing the risk of Cross-Site Request Forgery (CSRF) attacks.  4. \*\*Token Expiry:\*\* Implement token expiration to limit the duration of a session, forcing users to re-authenticate periodically.  5. \*\*Regenerate Tokens:\*\* Regenerate session tokens after login and periodically during a session to minimize the risk of session hijacking.  6. \*\*Store Minimal Information:\*\* Keep the cookie content minimal and do not store sensitive user information directly in the cookie.  7. \*\*Encryption:\*\* Encrypt the token to add an additional layer of security.  8. \*\*Validate Tokens:\*\* Ensure tokens are validated on the server side for each request to verify the user's role and permissions.  By adhering to these practices, you can significantly enhance the security of session management in your application.  There are four user roles with specific permissions:  1. \*\*Admin\*\*: Responsible for administrative and IT tasks. Has access to all documents and data across all classes and subjects.  2. \*\*Principal\*\*: Has read-only access to all exams and grades, and can approve them. Can generate reports to get an overview of each class.  3. \*\*Teachers\*\*: Can only view and manage documents related to their own class or subject.  4. \*\*Students\*\*: Can only view documents related to their own class or subject.  Additionally, users are grouped by different subjects to streamline access and management.  A secure password reset or account recovery process typically involves the following steps:  1. User initiates the password reset by clicking on a "Forgot Password" link.  2. User is prompted to enter their registered email address or username.  3. A verification code or password reset link is sent to the user's registered email address or phone number.  4. User clicks on the link or enters the verification code to access the password reset page.  5. User is prompted to answer security questions or complete additional verification steps, such as two-factor authentication, to confirm their identity.  6. Once verified, the user is allowed to create a new password.  7. User enters the new password and confirms it by entering it again.  8. User clicks a button to save the new password.  9. A confirmation message is sent to the user's registered email address or phone number, informing them of the password change.  10. The user can now log in with the new password.  The web application comprises several microservices, including:  1. User Administration  2. Authentication  3. OCR (Optical Character Recognition for handwritten letters)  4. Automatic Rating using Azure OpenAI  5. Prompt Management  6. Admin  Each of these microservices is deployed in a separate container within a Kubernetes cluster.  Application Architecture  Python Django as the middle layer, ReactJS for the frontend, Azure SQL Database (MySQL, Postgres, etc.), and Azure Blob Storage for storing scanned files  The frontend utilizes JavaScript to create a single-page application that serves as a dashboard for teachers. This dashboard includes login and logout functionality, allowing teachers to upload materials easily.  For the midlayer, we are using Apache Web Server to deploy a Python Django web application. The Django application handles all requests from the frontend and manages the workflow. It interacts with various microservices and external services such as Azure OpenAI and OCR Service. Additionally, the Django application accesses the database and blob storage.  To ensure the security of our Azure Database service, which stores user administration information, status information, prompts, and temporary processes, we should consider the following countermeasures:  1. \*\*SQL Injection Prevention:\*\* Use parameterized queries and stored procedures to prevent SQL injection attacks. Regularly update and patch database management systems.  2. \*\*Access Control:\*\* Implement strict access controls using Azure's role-based access control (RBAC). Ensure only authorized personnel have access to the database.  3. \*\*Encryption:\*\* Utilize Azure's built-in encryption at rest and in transit. Ensure data is encrypted using strong encryption standards like AES-256.  4. \*\*Audit Logging:\*\* Enable auditing to log access and changes to the database. Use Azure Monitor and Azure Security Center to track and respond to suspicious activities.  5. \*\*Firewall Configuration:\*\* Configure Azure SQL Database firewalls to limit access to trusted IP addresses only.  6. \*\*Regular Backups:\*\* Perform regular automated backups and ensure they are stored securely. Test backup restoration procedures periodically.  7. \*\*Input Validation:\*\* Validate all input data on the server side to prevent XSS, CSRF, and other injection attacks.  8. \*\*Compliance:\*\* Ensure compliance with relevant data protection regulations and standards such as GDPR, HIPAA, and SOC 2 by configuring and managing Azure Policy and Blueprints.  9. \*\*Security Updates:\*\* Regularly apply security patches and updates to the database service to mitigate known vulnerabilities.  10. \*\*OWASP Top 10:\*\* Regularly review and address the OWASP Top 10 security risks, ensuring that the application and database are safeguarded against common vulnerabilities such as broken authentication, sensitive data exposure, and security misconfigurations.  It is a 3-tier architecture consisting of the static layer, middle layer, and backend layer (data layer), with isolation between the different layers.  Biblioteca para autenticação, validação de segurança e gerenciamento de serviços de IA  Cloud Architecture  - Azure Functions: Implement network restrictions, use managed identities, enable authentication and authorization, enforce HTTPS, and apply role-based access control (RBAC).  - Application Gateway: Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), and enable diagnostic logging.  - Firewall: Configure application rules and network rules, enable threat intelligence-based filtering, implement logging and monitoring, and apply strict access control policies.  - Database: Enable encryption at rest and in transit, use automatic backups, implement Advanced Threat Protection, enforce strong authentication and access control, and regularly update and patch.  - Blob Storage: Enable encryption at rest, use private endpoints, enforce access policies and Shared Access Signatures (SAS), enable logging and monitoring, and configure data retention policies.  - Azure Kubernetes Service: Implement network policies, use managed identities, enable RBAC, configure pod security policies, regularly update and patch the cluster, and use Azure Policy for enforcement.  - Azure OpenAI Service: Enable network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, and ensure data encryption in transit and at rest.  - Azure AI Computer Vision: Restrict access using NSGs, enable authentication and authorization, monitor and log access, enforce data encryption in transit, and apply RBAC for resource access.  The relationship between these components and recommended security best practices are as follows:  - Azure Functions: Implement network restrictions, use managed identities, enable authentication and authorization, enforce HTTPS, and apply role-based access control (RBAC).  - Application Gateway: Enable Web Application Firewall (WAF), configure SSL/TLS termination, restrict access using Network Security Groups (NSGs), and enable diagnostic logging.  - Firewall: Configure application rules and network rules, enable threat intelligence-based filtering, implement logging and monitoring, and apply strict access control policies.  - Database: Enable encryption at rest and in transit, use automatic backups, implement Advanced Threat Protection, enforce strong authentication and access control, and regularly update and patch.  - Blob Storage: Enable encryption at rest, use private endpoints, enforce access policies and Shared Access Signatures (SAS), enable logging and monitoring, and configure data retention policies.  - Azure Kubernetes Service: Implement network policies, use managed identities, enable RBAC, configure pod security policies, regularly update and patch the cluster, and use Azure Policy for enforcement.  - Azure OpenAI Service: Enable network security, use managed identities, apply strict access control, monitor usage with diagnostic logs, and ensure data encryption in transit and at rest.  - Azure AI Computer Vision: Restrict access using NSGs, enable authentication and authorization, monitor and log access, enforce data encryption in transit, and apply RBAC for resource access.  A pure cloud environment without any connection to the school’s own environment is secure. The preferred architecture would be a fully cloud-based solution.  Access from the internet is direct. Evaluate if it meets the security classification.  The provided monitoring solutions are insufficient and should be integrated with the company’s central monitoring and SIEM solutions.  The current cloud governance model, managed by the school’s IT Department, is a good starting point. However, to ensure it is sufficient, consider enhancing it by implementing regular audits, establishing clear policies and procedures, incorporating automated compliance checks, and providing ongoing training for staff. |