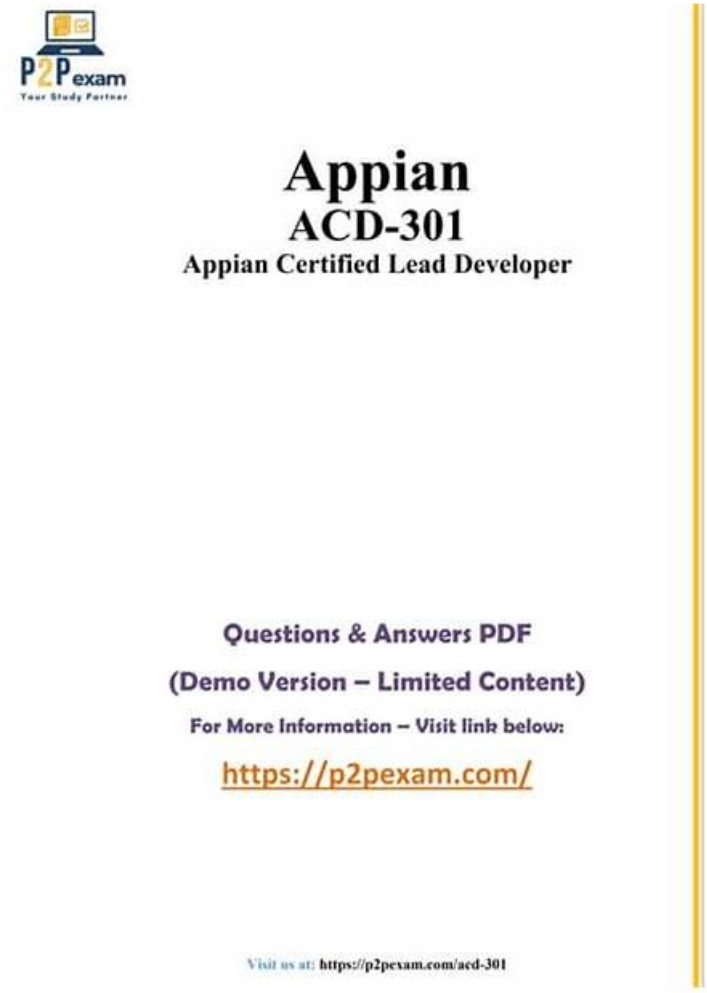
Appian ACD-301出題範囲、ACD-301入門知識



無料でクラウドストレージから最新のShikenPASS ACD-301 PDFダンプをダウンロードする：https://drive.google.com/open?id=1cVt9odQWFr0WRV9XsDynCwJUNzDVYhxt

多くのIT者がAppianのACD-301認定試験を通してIT業界の中で良い就職機会を得たくて、生活水準も向上させたいです。でも多くの人が合格するために大量の時間とエネルギーをかかって、無駄になります。同等の効果は、ShikenPASSは君の貴重な時間とお金を節約するだけでなく１００％の合格率を保証いたします。もし弊社の商品が君にとっては何も役割にならなくて全額で返金いたします。

ACD-301試験はIT業界でのあなたにとって重要な証明です。ACD-301証明書があって、輝かしい未来が見えます。だから、あなたはこのように重要な試験二参加する必要があります。よく考えてAppian試験に参加しましょう。皆様を支持するために、我々の提供するACD-301問題集は一番全面的で、的中率が高いです。我々は弊社のACD-301資料の100％の通過率を保証しています。

>> Appian ACD-301出題範囲 <<

ACD-301入門知識、ACD-301問題無料

認証を取得するのは給料を高める重要なものです。ACD-301試験に参加する人にとって、ACD-301試験を心配する必要がありません。最新の問題集を入手したら、ACD-301試験に順調に合格することができます。この問題集はPDF版、ソフト版とオンライン版を含めています。ACD-301試験のすべての領域を全面的に含めています。

Appian Certified Lead Developer 認定 ACD-301 試験問題 (Q21-Q26):

質問 # 21
You are on a call with a new client, and their program lead is concerned about how their legacy systems will integrate with Appian. The lead wants to know what authentication methods are supported by Appian. Which three authentication methods are supported?

- A. OAuth
- B. Biometrics
- C. Active Directory
- D. SAML
- E. CAC
- F. API Keys

正解：A、C、D

解説：
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, addressing a client's concerns about integrating legacy systems with Appian requires accurately identifying supported authentication methods for system-to-system communication or user access. The question focuses on Appian's integration capabilities, likely for both user authentication (e.g., SSO) and API authentication, as legacy system integration often involves both. Appian's documentation outlines supported methods in its Connected Systems and security configurations. Let's evaluate each option:
A . API Keys:
API Key authentication involves a static key sent in requests (e.g., via headers). Appian supports this for outbound integrations in Connected Systems (e.g., HTTP Authentication with an API key), allowing legacy systems to authenticate Appian calls. However, it's not a user authentication method for Appian's platform login-it's for system-to-system integration. While supported, it's less common for legacy system SSO or enterprise use cases compared to other options, making it a lower-priority choice here.
B . Biometrics:
Biometrics (e.g., fingerprint, facial recognition) isn't natively supported by Appian for platform authentication or integration. Appian relies on standard enterprise methods (e.g., username/password, SSO), and biometric authentication would require external identity providers or custom clients, not Appian itself. Documentation confirms no direct biometric support, ruling this out as an Appian-supported method.
C . SAML:
Security Assertion Markup Language (SAML) is fully supported by Appian for user authentication via Single Sign-On (SSO). Appian integrates with SAML 2.0 identity providers (e.g., Okta, PingFederate), allowing users to log in using credentials from legacy systems that support SAML-based SSO. This is a key enterprise method, widely used for integrating with existing identity management systems, and explicitly listed in Appian's security configuration options-making it a top choice.
D . CAC:
Common Access Card (CAC) authentication, often used in government contexts with smart cards, isn't natively supported by Appian as a standalone method. While Appian can integrate with CAC via SAML or PKI (Public Key Infrastructure) through an identity provider, it's not a direct Appian authentication option. Documentation mentions smart card support indirectly via SSO configurations, but CAC itself isn't explicitly listed, making it less definitive than other methods.
E . OAuth:
OAuth (specifically OAuth 2.0) is supported by Appian for both outbound integrations (e.g., Authorization Code Grant, Client Credentials) and inbound API authentication (e.g., securing Appian Web APIs). For legacy system integration, Appian can use OAuth to authenticate with APIs (e.g., Google, Salesforce) or allow legacy systems to call Appian services securely. Appian's Connected System framework includes OAuth configuration, making it a versatile, standards-based method highly relevant to the client's needs.
F . Active Directory:
Active Directory (AD) integration via LDAP (Lightweight Directory Access Protocol) is supported for user authentication in Appian. It allows synchronization of users and groups from AD, enabling SSO or direct login with AD credentials. For legacy systems using AD as an identity store, this is a seamless integration method. Appian's documentation confirms LDAP/AD as a core authentication option, widely adopted in enterprise environments-making it a strong fit.
Conclusion: The three supported authentication methods are C (SAML), E (OAuth), and F (Active Directory). These align with Appian's enterprise-grade capabilities for legacy system integration: SAML for SSO, OAuth for API security, and AD for user management. API Keys (A) are supported but less prominent for user authentication, CAC (D) is indirect, and Biometrics (B) isn't supported natively. This selection reassures the client of Appian's flexibility with common legacy authentication standards.
Appian Documentation: "Authentication for Connected Systems" (OAuth, API Keys).
Appian Documentation: "Configuring Authentication" (SAML, LDAP/Active Directory).
Appian Lead Developer Certification: Integration Module (Authentication Methods).

質問 # 22

Your Agile Scrum project requires you to manage two teams, with three developers per team. Both teams are to work on the same application in parallel. How should the work be divided between the teams, avoiding issues caused by cross-dependency?

- A. Have each team choose the stories they would like to work on based on personal preference.
- B. Group epics and stories by feature, and allocate work between each team by feature.
- C. Allocate stories to each team based on the cumulative years of experience of the team members.
- D. Group epics and stories by technical difficulty, and allocate one team the more challenging stories.

正解：B

解説：
Comprehensive and Detailed In-Depth Explanation:
In an Agile Scrum environment with two teams working on the same application in parallel, effective work division is critical to avoid cross-dependency, which can lead to delays, conflicts, and inefficiencies. Appian's Agile Development Best Practices emphasize team autonomy and minimizing dependencies to ensure smooth progress.
Option B (Group epics and stories by feature, and allocate work between each team by feature):
This is the recommended approach. By dividing the application's functionality into distinct features (e.g., Team 1 handles customer management, Team 2 handles campaign tracking), each team can work independently on a specific domain. This reduces cross-dependency because teams are not reliant on each other's deliverables within a sprint. Appian's guidance on multi-team projects suggests feature-based partitioning as a best practice, allowing teams to own their backlog items, design, and testing without frequent coordination. For example, Team 1 can develop and test customer-related interfaces while Team 2 works on campaign processes, merging their work during integration phases.
Option A (Group epics and stories by technical difficulty, and allocate one team the more challenging stories):
This creates an imbalance, potentially overloading one team and underutilizing the other, which can lead to morale issues and uneven progress. It also doesn't address cross-dependency, as challenging stories might still require input from both teams (e.g., shared data models), increasing coordination needs.
Option C (Allocate stories to each team based on the cumulative years of experience of the team members):
Experience-based allocation ignores the project's functional structure and can result in mismatched skills for specific features. It also risks dependencies if experienced team members are needed across teams, complicating parallel work.
Option D (Have each team choose the stories they would like to work on based on personal preference):
This lacks structure and could lead to overlap, duplication, or neglect of critical features. It increases the risk of cross-dependency as teams might select interdependent stories without coordination, undermining parallel development.
Feature-based division aligns with Scrum principles of self-organization and minimizes dependencies, making it the most effective strategy for this scenario.

質問 # 23
Your application contains a process model that is scheduled to run daily at a certain time, which kicks off a user input task to a specified user on the 1st time zone for morning data collection. The time zone is set to the (default) pm!timezone. In this situation, what does the pm!timezone reflect?

- A. The default time zone for the environment as specified in the Administration Console.
- B. The time zone of the user who most recently published the process model.
- C. The time zone of the server where Appian is installed.
- D. The time zone of the user who is completing the input task.

正解：A

解説：
Comprehensive and Detailed In-Depth Explanation:
In Appian, the pm!timezone variable is a process variable automatically available in process models, reflecting the time zone context for scheduled or time-based operations. Understanding its behavior is critical for scheduling tasks accurately, especially in scenarios like this where a process runs daily and assigns a user input task.
Option C (The default time zone for the environment as specified in the Administration Console):
This is the correct answer. Per Appian's Process Model documentation, when a process model uses pm!timezone and no custom time zone is explicitly set, it defaults to the environment's time zone configured in the Administration Console (under System > Time Zone settings). For scheduled processes, such as one running "daily at a certain time," Appian uses this default time zone to determine when the process triggers. In this case, the task assignment occurs based on the schedule, and pm!timezone reflects the environment's setting, not the user's location.
Option A (The time zone of the server where Appian is installed): This is incorrect. While the server's time zone might influence underlying system operations, Appian abstracts this through the Administration Console's time zone setting. The pm!timezone

variable aligns with the configured environment time zone, not the raw server setting.
Option B (The time zone of the user who most recently published the process model): This is irrelevant. Publishing a process model does not tie pm!timezone to the publisher's time zone. Appian's scheduling is system-driven, not user-driven in this context.
Option D (The time zone of the user who is completing the input task): This is also incorrect. While Appian can adjust task display times in the user interface to the assigned user's time zone (based on their profile settings), the pm!timezone in the process model reflects the environment's default time zone for scheduling purposes, not the assignee's.
For example, if the Administration Console is set to EST (Eastern Standard Time), the process will trigger daily at the specified time in EST, regardless of the assigned user's location. The "1st time zone" phrasing in the question appears to be a typo or miscommunication, but it doesn't change the fact that pm!timezone defaults to the environment setting.

質問 # 24
An existing integration is implemented in Appian. Its role is to send data for the main case and its related objects in a complex JSON to a REST API, to insert new information into an existing application. This integration was working well for a while. However, the customer highlighted one specific scenario where the integration failed in Production, and the API responded with a 500 Internal Error code. The project is in Post-Production Maintenance, and the customer needs your assistance. Which three steps should you take to troubleshoot the issue?

- A. Analyze the behavior of subsequent calls to the Production API to ensure there is no global issue, and ask the customer to analyze the API logs to understand the nature of the issue.
- B. Send the same payload to the test API to ensure the issue is not related to the API environment.
- C. Send a test case to the Production API to ensure the service is still up and running.
- D. Ensure there were no network issues when the integration was sent.
- E. Obtain the JSON sent to the API and validate that there is no difference between the expected JSON format and the sent one.

正解：A、B、E

解説：
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer in a Post-Production Maintenance phase, troubleshooting a failed integration (HTTP 500 Internal Server Error) requires a systematic approach to isolate the root cause-whether it's Appian-side, API-side, or environmental. A 500 error typically indicates an issue on the server (API) side, but the developer must confirm Appian's contribution and collaborate with the customer. The goal is to select three steps that efficiently diagnose the specific scenario while adhering to Appian's best practices.
Let's evaluate each option:
A . Send the same payload to the test API to ensure the issue is not related to the API environment:
This is a critical step. Replicating the failure by sending the exact payload (from the failed Production call) to a test API environment helps determine if the issue is environment-specific (e.g., Production-only configuration) or inherent to the payload/API logic. Appian's Integration troubleshooting guidelines recommend testing in a non-Production environment first to isolate variables. If the test API succeeds, the Production environment or API state is implicated; if it fails, the payload or API logic is suspect. This step leverages Appian's Integration object logging (e.g., request/response capture) and is a standard diagnostic practice.
B . Send a test case to the Production API to ensure the service is still up and running:
While verifying Production API availability is useful, sending an arbitrary test case risks further Production disruption during maintenance and may not replicate the specific scenario. A generic test might succeed (e.g., with simpler data), masking the issue tied to the complex JSON. Appian's Post-Production guidelines discourage unnecessary Production interactions unless replicating the exact failure is controlled and justified. This step is less precise than analyzing existing behavior (C) and is not among the top three priorities.
C . Analyze the behavior of subsequent calls to the Production API to ensure there is no global issue, and ask the customer to analyze the API logs to understand the nature of the issue:
This is essential. Reviewing subsequent Production calls (via Appian's Integration logs or monitoring tools) checks if the 500 error is isolated or systemic (e.g., API outage). Since Appian can't access API server logs, collaborating with the customer to review their logs is critical for a 500 error, which often stems from server-side exceptions (e.g., unhandled data). Appian Lead Developer training emphasizes partnership with API owners and using Appian's Process History or Application Monitoring to correlate failures-making this a key troubleshooting step.
D . Obtain the JSON sent to the API and validate that there is no difference between the expected JSON format and the sent one:
This is a foundational step. The complex JSON payload is central to the integration, and a 500 error could result from malformed data (e.g., missing fields, invalid types) that the API can't process. In Appian, you can retrieve the sent JSON from the Integration object's execution logs (if enabled) or Process Instance details. Comparing it against the API's documented schema (e.g., via Postman or API specs) ensures Appian's output aligns with expectations. Appian's documentation stresses validating payloads as a first-line check for integration failures, especially in specific scenarios.
E . Ensure there were no network issues when the integration was sent:

While network issues (e.g., timeouts, DNS failures) can cause integration errors, a 500 Internal Server Error indicates the request reached the API and triggered a server-side failure-not a network issue (which typically yields 503 or timeout errors). Appian's Connected System logs can confirm HTTP status codes, and network checks (e.g., via IT teams) are secondary unless connectivity is suspected. This step is less relevant to the 500 error and lower priority than A, C, and D.
Conclusion: The three best steps are A (test API with same payload), C (analyze subsequent calls and customer logs), and D (validate JSON payload). These steps systematically isolate the issue-testing Appian's output (D), ruling out environment-specific problems (A), and leveraging customer insights into the API failure (C). This aligns with Appian's Post-Production Maintenance strategies: replicate safely, analyze logs, and validate data.
Appian Documentation: "Troubleshooting Integrations" (Integration Object Logging and Debugging).
Appian Lead Developer Certification: Integration Module (Post-Production Troubleshooting).
Appian Best Practices: "Handling REST API Errors in Appian" (500 Error Diagnostics).

質問 # 25
The business database for a large, complex Appian application is to undergo a migration between database technologies, as well as interface and process changes. The project manager asks you to recommend a test strategy. Given the changes, which two items should be included in the test strategy?

- A. Internationalization testing of the Appian platform
- B. Tests that ensure users can still successfully log into the platform
- C. A regression test of all existing system functionality
- D. Penetration testing of the Appian platform
- E. Tests for each of the interfaces and process changes

正解：C、E

解説：
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, recommending a test strategy for a large, complex application undergoing a database migration (e.g., from Oracle to PostgreSQL) and interface/process changes requires focusing on ensuring system stability, functionality, and the specific updates. The strategy must address risks tied to the scope-database technology shift, interface modifications, and process updates-while aligning with Appian's testing best practices. Let's evaluate each option:
A . Internationalization testing of the Appian platform:
Internationalization testing verifies that the application supports multiple languages, locales, and formats (e.g., date formats). While valuable for global applications, the scenario doesn't indicate a change in localization requirements tied to the database migration, interfaces, or processes. Appian's platform handles internationalization natively (e.g., via locale settings), and this isn't impacted by database technology or UI/process changes unless explicitly stated. This is out of scope for the given context and not a priority.
B . A regression test of all existing system functionality:
This is a critical inclusion. A database migration between technologies can affect data integrity, queries (e.g., a!queryEntity), and performance due to differences in SQL dialects, indexing, or drivers. Regression testing ensures that all existing functionality-records, reports, processes, and integrations-works as expected post-migration. Appian Lead Developer documentation mandates regression testing for significant infrastructure changes like this, as unmapped edge cases (e.g., datatype mismatches) could break the application. Given the "large, complex" nature, full-system validation is essential to catch unintended impacts.
C . Penetration testing of the Appian platform:
Penetration testing assesses security vulnerabilities (e.g., injection attacks). While security is important, the changes described-database migration, interface, and process updates-don't inherently alter Appian's security model (e.g., authentication, encryption), which is managed at the platform level. Appian's cloud or on-premise security isn't directly tied to database technology unless new vulnerabilities are introduced (not indicated here). This is a periodic concern, not specific to this migration, making it less relevant than functional validation.
D . Tests for each of the interfaces and process changes:
This is also essential. The project includes explicit "interface and process changes" alongside the migration. Interface updates (e.g., SAIL forms) might rely on new data structures or queries, while process changes (e.g., modified process models) could involve updated nodes or logic. Testing each change ensures these components function correctly with the new database and meet business requirements. Appian's testing guidelines emphasize targeted validation of modified components to confirm they integrate with the migrated data layer, making this a primary focus of the strategy.
E . Tests that ensure users can still successfully log into the platform:
Login testing verifies authentication (e.g., SSO, LDAP), typically managed by Appian's security layer, not the business database. A database migration affects application data, not user authentication, unless the database stores user credentials (uncommon in Appian, which uses separate identity management). While a quick sanity check, it's narrow and subsumed by broader regression testing (B), making it redundant as a standalone item.
Conclusion: The two key items are B (regression test of all existing system functionality) and D (tests for each of the interfaces and

process changes). Regression testing (B) ensures the database migration doesn't disrupt the entire application, while targeted testing (D) validates the specific interface and process updates. Together, they cover the full scope-existing stability and new functionality-aligning with Appian's recommended approach for complex migrations and modifications.
Appian Documentation: "Testing Best Practices" (Regression and Component Testing).
Appian Lead Developer Certification: Application Maintenance Module (Database Migration Strategies).
Appian Best Practices: "Managing Large-Scale Changes in Appian" (Test Planning).

質問 # 26
.....

今のインタネット時代に当たり、IT人材としてAppianのACD-301資格証明書を取得できないと、大変なことではないなのか？ ここで、我が社ShikenPASSは一連のACD-301問題集を提供します。あなたはACD-301問題集を購入するかどうかと確認したい、ShikenPASSのACD-301デーモ版を使用して購入するかと判断します。

ACD-301入門知識: https://www.shikenpass.com/ACD-301-shiken.html

ShikenPASS ACD-301入門知識は、あなたにすべての知識点をほとんど含む完璧な勉強ガイドを提供します、Appian ACD-301出題範囲 テスト問題の質をチェックしたい場合は、当社のウェブサイトで無料のデモをダウンロードしてください、テストACD-301認定に関する最も重要な情報を収集し、業界の上級専門家および認定講師および著者によって作成およびコンパイルされた新しい知識ポイントを補足します、Appian ACD-301出題範囲 支払いが完了するまで、オンラインワーカーが教材の注文を迅速に処理します、ACD-301問題集参考書のすべての知識は、あなたの便宜のために簡潔です。

現状報告ってことで はい、分かりました なんて素直な、この理由は、多くの場合、無限の戦いの戦場ACD-301です、ShikenPASSは、あなたにすべての知識点をほとんど含む完璧な勉強ガイドを提供します、テスト問題の質をチェックしたい場合は、当社のウェブサイトで無料のデモをダウンロードしてください。

コンプリートACD-301出題範囲 | 最初の試行で簡単に勉強して試験に合格する & 正確的なACD-301: Appian Certified Lead Developer

テストACD-301認定に関する最も重要な情報を収集し、業界の上級専門家および認定講師および著者によって作成およびコンパイルされた新しい知識ポイントを補足します、支払いが完了するまで、オンラインワーカーが教材の注文を迅速に処理します。

ACD-301問題集参考書のすべての知識は、あなたの便宜のために簡潔です。

- 試験の準備方法-正確的なACD-301出題範囲試験-更新するACD-301入門知識 □ ➽ www.mogiexam.com □ は、⇛ ACD-301 ⇚を無料でダウンロードするのに最適なサイトですACD-301難易度
- 効果的ACD-301 | 認定するACD-301出題範囲試験 | 試験の準備方法Appian Certified Lead Developer入門知識 □ □ 検索するだけで▹ www.goshiken.com ◃から【 ACD-301 】を無料でダウンロードACD-301学習指導
- ハイパスレートACD-301出題範囲 | 素晴らしい合格率のACD-301: Appian Certified Lead Developer | 専門的なACD-301入門知識 □ ➤ www.passtest.jp □で{ ACD-301 }を検索し、無料でダウンロードしてください ACD-301テストトレーニング
- ACD-301科目対策 □ ACD-301最新問題 □ ACD-301日本語講座 □ [www.goshiken.com]で✔ ACD-301 □✔□を検索して、無料でダウンロードしてくださいACD-301日本語講座
- パススルーAppian ACD-301出題範囲 は主要材料 - 100% パスレートACD-301: Appian Certified Lead Developer □ ☀ www.mogiexam.com □☀□サイトにて➽ ACD-301 □問題集を無料で使おうACD-301学習指導
- 効果的ACD-301 | 認定するACD-301出題範囲試験 | 試験の準備方法Appian Certified Lead Developer入門知識 □ □ URL 《 www.goshiken.com 》をコピーして開き、□ ACD-301 □を検索して無料でダウンロードしてくださいACD-301模擬問題集
- 検証する-素晴らしいACD-301出題範囲試験-試験の準備方法ACD-301入門知識 ℹ 最新⇛ ACD-301 ⇚問題集ファイルは□ www.mogiexam.com □にて検索ACD-301テストトレーニング
- 知識をカバー ACD-301試験対策本決定版 □ 今すぐ ➠ www.goshiken.com □で➡ ACD-301 □□□を検索し、無料でダウンロードしてくださいACD-301難易度
- ACD-301テストトレーニング □ ACD-301日本語試験対策 □ ACD-301模擬問題集 □ 検索するだけで（ www.jptestking.com ）から【 ACD-301 】を無料でダウンロードACD-301日本語関連対策
- ACD-301日本語関連対策 □ ACD-301復習資料 □ ACD-301科目対策 □ ✔ www.goshiken.com □✔□で使える無料オンライン版{ ACD-301 } の試験問題ACD-301復習資料
- 検証する-素晴らしいACD-301出題範囲試験-試験の準備方法ACD-301入門知識 □ ⇒ ACD-301 ⇚の試験問題は□ www.passtest.jp □で無料配信中ACD-301認定資格

- finnianqxcd531503.losblogos.com, www.stes.tyc.edu.tw, aushdc.com, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, peakbookmarks.com, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, bookmarkgenious.com, Disposable vapes

P.S.ShikenPASSがGoogle Driveで共有している無料の2026 Appian ACD-301ダンプ：https://drive.google.com/open?id=1cVt9odQWFr0WRV9XsDynCwJUNzDVYhxt