SAP
C_FIORD
SAP Certified - SAP Fiori Application Developer

For More Information – Visit link below:
https://www.examsempire.com/

Product Version

1. **Up to Date products, reliable and verified.**
2. **Questions and Answers in PDF Format.**

Latest Version: 4.0

1. Micro Skill Drill Exam
2. Unified Scenario Exam

Topic: 1
Micro Skill Drill Exam

Question: 1

An agricultural cooperative is planning a SAP Fiori application for seed allocation request review. Allocation planners need to filter requests, open a selected request, review grower details and requested quantities, update a simple allocation decision, and save the reviewed object. The application will consume an OData V4 service, and the first release must support structured request processing rather than executive dashboard monitoring.
One developer proposes an overview-style application because leadership may later ask for allocation summary cards. Another proposes using a standard list-to-object SAP Fiori elements pattern for the confirmed review flow and reserving overview content for a later requirement. The constraint is that the first release must avoid over-design while still supporting the complete planner review process.
Which recommendation best fits the first-release objective?
Response:

A. Use an overview-style application now so future allocation summary cards can be added without changing direction.
B. Use a standard list-to-object SAP Fiori elements pattern for the allocation review flow and defer overview content until confirmed.
C. Build a freestyle SAPUI5 application because it can support both request processing and future dashboard needs with full control.
D. Deliver only a searchable request list first and postpone allocation decision updates until a later release.

Answer: B

Explanation:
Feedback:
This option aligns the application pattern with the confirmed processing flow. It supports filtering, opening, reviewing, updating, and saving allocation requests while avoiding premature design for unconfirmed dashboard content.

Question: 2

A fleet maintenance company is preparing a SAPUI5 application for vehicle inspection review. The application runs in a platform-based development environment and retrieves inspection result, mileage, and next-service date from an OData service. During testing, inspectors confirm that mileage updates

correctly when a different vehicle is selected, but the next-service message still reflects the previously selected vehicle.
The developer finds that the message is built once in controller logic during initial load, while the mileage field uses binding to the selected vehicle context. One teammate suggests recalculating the message manually after every selection change. Another suggests binding the message content to the selected context and moving translatable text to the localization layer. The constraint is that additional inspection messages will be added later with minimal controller changes.
Which corrective direction should the developer recommend?
Response:

A. Recalculate the next-service message manually in controller logic whenever the selected vehicle changes.
B. Bind the message content to the selected vehicle context and use localization resources for user-facing text.
C. Reload the inspection review page after every vehicle selection so all messages are rebuilt from initial startup.
D. Remove the next-service message and rely only on the next-service date field because mileage already updates correctly.

Answer: B

Explanation:
Feedback:
This option addresses the issue at the binding and localization layers. Binding message content to the selected vehicle context keeps displayed values current, while localization resources keep user-facing text maintainable as additional messages are introduced.

Question: 3

A regional renewable-power operator is building a SAP Fiori application for maintenance permit review. The frontend consumes an OData V4 service that exposes permit requests, planned work date, and service-confirmed urgency ranking. In the prototype, the list displays permit records correctly, but sorting by planned work date gives inconsistent results because the controller sorts formatted date text instead of the service-exposed date value.
The project lead wants sorting to remain reliable when user locale settings change. One developer proposes adding locale-specific text parsing in controller logic. Another proposes binding the sort behavior to the service-exposed date value and validating the returned order against permit dat
a. The constraint is that additional date-based review fields will be added later, and sorting must remain service-consistent.
Which recommendation best addresses the sorting issue?
Response:
A. Bind sorting to the service-exposed date value and validate the returned order against permit data.
B. Add locale-specific text parsing in controller logic so each formatted date string can be sorted correctly.
C. Remove date sorting from the first release because permit requests already display in the list.
D. Ask users to export permit records and sort planned work dates outside the SAP Fiori application.

Answer: A

Explanation:
Feedback:
This option addresses the issue at the service-value and binding layer. Sorting against the service-exposed date value keeps ordering stable across locale-specific display formats and supports validation against the returned permit data.

Question: 4

A regional exhibition-services company is extending a SAP Fiori application for booth-service request review. The team uses SAP Business Application Studio and already has a project connected to an OData V4 service for service requests, exhibitor details, and review status. The product owner now wants to add a field to the existing request review page showing whether exhibitor setup documents are complete.
One developer suggests directly editing the page artifacts and binding expression by hand so the walkthrough can be updated quickly. Another suggests using guided development to add the service-exposed field where possible, then reviewing the generated configuration and affected bindings before the walkthrough. The constraint is that more document-related fields will be added later, and the team wants the extension work to remain traceable and consistent.
Which recommendation best supports controlled field enhancement?
Response:

A. Directly edit the page artifacts and binding expression by hand so the new field appears quickly.
B. Copy an existing displayed field and change its visible label so the walkthrough can show setup-document status.
C. Use guided development to add the service-exposed field, then review generated configuration and affected bindings.
D. Defer the setup-document field until all future document-related fields are finalized by the product owner.

Answer: C

Explanation:
Feedback:
This option supports enhancement at the development-tooling and service-artifact layer. Guided addition followed by configuration and binding review keeps the field aligned with the OData V4 service while preserving developer validation.

Question: 5

An engineering services firm is preparing a SAPUI5 project-cost dashboard for release to multiple regional teams. The dashboard must show project values from an OData service and display labels, messages, and status text in the user's language. In the current prototype, numerical values update correctly when the service changes, but translated status text is being assembled inside controller logic using conditional statements.

The product owner wants the first release completed quickly, but the application must support additional languages without controller rewrites. One developer suggests leaving the controller logic because it already produces the correct English and German text. Another suggests moving translatable text to the localization mechanism and keeping project values bound to the service model. The constraint is maintainable multilingual behavior without disturbing the working data binding.
Which corrective action should the developer recommend?
Response:

A. Keep the controller-based text logic because it already handles the currently required languages and does not affect service values.
B. Store all translated status text as additional project fields in the OData service so one binding source controls both values and labels.
C. Move translatable UI text to the localization mechanism while leaving project values bound to the OData service model.
D. Create a separate controller for each language so every regional team can maintain its own status text conditions independently.

Answer: C

Explanation:
Feedback:
This option corrects the issue at the appropriate application layer. Localization resources handle translatable labels, messages, and status text, while the OData model continues to provide project values through data binding.

Question: 6

A retail buying group is creating a SAPUI5 application for supplier onboarding status. The application runs in a platform-based development landscape and must load consistently from the launchpad with the correct application descriptor, component initialization, and service model configuration. During review, one prototype opens locally from the development workspace, but the same application fails to load reliably when started through the intended launchpad tile.
A developer suggests adjusting the first XML view because the screen layout is where users see the failure. Another developer recommends reviewing the application descriptor and component startup configuration before changing view content. The project constraint is that the application must have a stable entry point for later additions such as localized text and additional service models.
What is the best recommendation for resolving the startup issue?
Response:

A. Redesign the first XML view so it can initialize service data directly when the launchpad tile opens the application.
B. Review the application descriptor and component startup configuration so the app has a stable launch and model initialization path.
C. Add a controller method to each view that checks whether the launchpad started the application before rendering screen content.
D. Keep the local workspace startup behavior and instruct users to open the application only from the development preview until the next release.

Answer: B

Explanation:
Feedback:
This option addresses the correct initialization layer. The application descriptor and component startup configuration support consistent launch behavior, model setup, and later addition of shared resources such as localization and additional service models.

Question: 7

A specialty food importer is enhancing a SAP Fiori elements application for import-lot compliance review. The existing application uses standard list and object behavior over an OData V4 service. Compliance analysts want an additional filter for inspection priority so they can narrow the list before opening import lots, but they do not want the existing list behavior, object navigation, or standard review flow replaced.
One developer proposes rebuilding the list page as a freestyle SAPUI5 view to control the filter area completely. Another proposes adding the filter through a supported extension or configuration-aligned approach while preserving the template-driven list behavior. The constraint is that the enhancement must be delivered quickly and should remain compatible with later template-aligned changes.
Which recommendation best satisfies the filter enhancement?
Response:

A. Add the inspection-priority filter through a supported extension or configuration-aligned approach while preserving template-driven list behavior.
B. Rebuild the list page as a freestyle SAPUI5 view so all filter behavior can be controlled outside the template.
C. Remove object navigation from the first release so analysts can focus only on filtered import-lot lists.
D. Ask analysts to filter inspection priority outside the application until the full review flow is redesigned.

Answer: A

Explanation:
Feedback:
This option satisfies the requested list-level enhancement while preserving the existing template-driven flow. It adds the needed filtering capability without taking ownership of standard list behavior and object navigation unnecessarily.

Question: 8

A regional public-works contractor is modernizing a legacy work-order acceptance screen into a SAPUI5 application. The application runs in a platform-based development environment and must initialize routing, a service model for work orders, and a shared view setting for compact review mode. In testing, the overview opens correctly, but a bookmarked work-order detail route opens with the compact setting missing until users return to the overview once.

One developer suggests adding compact-setting initialization inside the detail controller. Another suggests initializing shared settings at the application component level so all routes receive the same state regardless of entry path. The constraint is that more bookmarked work-order routes will be supported later, and each route must render consistently without depending on the overview being opened first.
What should the developer recommend?
Response:

A. Add compact-setting initialization inside the detail controller so the bookmarked route can repair its own state.
B. Disable bookmarked work-order detail routes so users always start from the overview where compact mode is available.
C. Create separate applications for overview and detail entry so each route type can initialize its own settings.
D. Initialize shared settings at the application component level so all routes receive consistent state before rendering.

Answer: D

Explanation:
Feedback:
This option addresses the issue at the application startup layer. Component-level initialization makes shared route settings available before dependent pages render, supporting consistent behavior across overview and bookmarked entry paths.

Question: 9

A regional legal process outsourcing company is preparing a SAPUI5 application for matter intake review. The application runs in a platform-based development environment and retrieves matter identifier, review stage, responsible analyst, and due date from an OData service. Reviewers can temporarily pin several matters for personal follow-up during a session, but pinning should not change the service-backed matter record.
During testing, pinning a matter triggers the same pending-change state used when the review stage is edited. One developer suggests suppressing pending-change checks when only a pin is changed. Another suggests storing pin selections in a separate UI state model while keeping matter data bound to the service model. The constraint is that more session-only review aids will be added later.
Which corrective direction should the developer recommend?
Response:

A. Suppress pending-change checks whenever only a temporary matter pin is changed.
B. Store temporary pin selections in a separate UI state model while keeping matter data service-bound.
C. Save pins into the matter record so the pending-change state accurately reflects a service-backed edit.
D. Remove pinning from the first release so reviewers cannot create temporary UI-only state.

Answer: B

Explanation:
Feedback:
This option addresses the issue at the state-management layer. Keeping temporary pins in a UI state model prevents personal review aids from being interpreted as changes to matter records.

Question: 10

A textile exporter is planning a SAP Fiori application for sales margin exception analysis. Managers need to review margin exceptions by customer group, compare aggregated margin impact, and drill into exception records when follow-up is required. The application will consume an OData V4 service, and the first release must support analysis-oriented review rather than routine record maintenance.
One developer proposes using a standard List Report because exception records can be filtered quickly. Another proposes selecting a SAP Fiori elements pattern better suited to analytical list behavior and drilldown, then using extensions only for gaps that remain after the standard pattern is evaluated. The constraint is that managers must identify business impact quickly without forcing the frontend team into full custom dashboard ownership.
Which recommendation best aligns with the application purpose?
Response:

A. Use a standard List Report because filtering exception records is enough for the first release.
B. Build a freestyle SAPUI5 dashboard immediately so every chart and aggregation can be controlled manually.
C. Choose a SAP Fiori elements pattern suited to analytical review and drilldown, then extend only where standard behavior is insufficient.
D. Deliver only a downloadable exception list and let managers perform margin analysis outside the SAP Fiori application.

Answer: C

Explanation:
Feedback:
This option aligns the application pattern with analytical review and controlled drilldown. It supports business-impact evaluation while limiting custom development to confirmed gaps in the standard pattern.
Topic: 2
Unified Scenario Exam

Question: 11

CHALLENGE 2 — Reusable Note Preview Behavior Across Entry Points
The note preview component works when embedded on the object page but behaves unpredictably when opened from a compact list-row action. What should the developer investigate first?
Response:

A. Whether the reusable component receives the required context and note references from both the object page and list-row entry point.
B. Whether the object page note preview should be removed so coordinators only use the list-row action.
C. Whether the note preview should be duplicated as separate custom UI code for each entry point.
D. Whether coordinators should return to shared inspection lists whenever note preview behavior is uncertain.

Answer: A

Explanation:
Feedback:
Different behavior by entry point points first to component context and note reference binding. This validates the reusable design before replacing or duplicating it.

Question: 12

CHALLENGE 2 — Reusable Note Preview Behavior Across Entry Points
Coordinators want faster access to quality notes, but the UX reviewer wants a consistent release review flow. Which recommendation best balances these priorities?
Response:

A. Add list-row note previews for every exception immediately, even if behavior differs from the object page.
B. Keep notes only on the object page because consistent flow is always more important than dispatch preparation speed.
C. Validate reusable note preview behavior across intended entry points, then enable faster access only where context and flow remain consistent.
D. Create a separate note preview application so coordinators can bypass the release-check object page entirely.

Answer: C

Explanation:
Feedback:
This balances faster note access with a consistent review flow. It allows the enhancement only after context and user flow are validated.

Question: 13

CHALLENGE 3 — Extension Boundary for Release Review Actions
The operations lead asks for faster release actions during dispatch preparation. The development lead wants to stay close to generated application behavior. What is the best recommendation?
Response:

A. Add broad controller logic immediately so every release action can be handled outside the generated pattern.
B. Remove release actions from the first wave and require coordinators to record outcomes in the old inspection list.
C. Validate whether supported configuration or extension mechanisms can handle the action behavior before adding broader custom controller logic.
D. Build a separate freestyle action page so coordinators can process release decisions without using the object page.

Answer: C

Explanation:
Feedback:
This checks the maintainable extension boundary before increasing custom logic. It supports operational speed while respecting the generated application direction.

Question: 14

CHALLENGE 3 — Extension Boundary for Release Review Actions
Quality teams previously used different labels for similar release outcomes. Which release action approach best supports the rollout-wave objective?
Response:

A. Allow each plant to define local action names so coordinators can preserve familiar review habits.
B. Use a common action design that supports the required release behavior and keeps outcome meaning consistent across plants.
C. Hide release actions during the first wave so local teams cannot disagree about labels.
D. Put all action handling into a custom controller and let each plant request its own behavior later.

Answer: B

Explanation:
Feedback:
A common action design supports consistent quality interpretation while still enabling the release process. It addresses the scenario’s cross-plant consistency concern.

Thank You for Trying Our Product

Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

Check our Customer Testimonials and ratings available on every product page.

Visit our website.

https://examsempire.com/