

SE-01-TS13

Technical Specification

MoodBoss backend enhancements: Assistant, Event Card, Calories and Emotion Try-On

Parameter	Value
Version	v1.4 - full edition without estimate
Date	30.04.2026
Status	Final technical statement for backend implementation of the first commercial release. The estimate is issued as a separate document.
Related documents	SE-01-TS11 - SUK MoodBoss; SE-01-SM13 - TS13 estimate; SUK-MP Dependency Matrix v2.1

1. Scope of Work

This Technical Specification defines the full scope of MoodBoss backend enhancements for the first commercial release in the following areas:

- hybrid Assistant model: quick buttons and launch of the same scenarios from a regular text request;
- Event Card as the core cross-cutting product entity;
- calorie scenario inside the Assistant, with the result saved to the Event Card and displayed in the Health section;
- Emotion Try-On as a scenario for working with the Event Card inside the Assistant;
- Assistant topics as separate chats / cases without a top-level project hierarchy;
- backend technical readiness for subsequent integration with SUK, without implementing a full SUK within this Technical Specification

This document does not include hours, cost, payment terms or the commercial schedule. These items are defined only in the separate estimate SE-01-SM13.

2. Objective of the Work

The objective is to transform the Assistant from a simple text chat and a set of separate buttons into a working MoodBoss scenario contour that helps the user move from a request to a saved structured product object.

The backend must provide:

- scenario launch through a quick button on the Assistant screen;
- launch of the same scenario through a regular text request in chat;
- user intent detection for key scenarios;
- clarifying questions when data is insufficient;
- structured actions for the frontend;
- creation of a draft / pending action;
- preview of the result before saving;
- saving of the Event Card, calorie result or Emotion Try-On result only after user confirmation;
- storage of the result as a structured backend object, not only as chat history.

3. Release Architectural Formula

Assistant -> button or text request -> intent understanding -> clarifying questions -> actions for frontend -> draft -> preview -> user confirmation -> EventCard -> Health / Calendar.

This formula is the basic implementation rule. No structured object may be saved silently without user confirmation if the matter concerns an Event Card, calories or an Emotion Try-On result.

4. Basic Implementation Principles

- A button and a text request lead to one backend scenario contour, not to separate independent implementations.
- The backend must return to the frontend not only a text response, but also structured blocks: quick replies, action buttons, preview and status block.
- EventCard is the central entity to which scenarios are linked if their result should become a fact of the user's day.

- Health displays aggregates and state data, including calorie balance, but the calorie scenario itself is launched through the Assistant.
- Calendar is the main lifecycle contour for the Event Card.
- Emotion Try-On is not a separate top-level section of the first release, it works as an Assistant scenario and as a linked EventCard result.
- Assistant topics are treated as separate chats / cases. Top-level Projects are outside the scope of the first stage.
- SUK, showcases, aggregates and management analytics are implemented under the separate Technical Specification SE-01-TS11. TS13 includes only backend technical readiness to transmit events and links.

5. Assistant: Hybrid Model

5.1. General Concept

The hybrid Assistant model is confirmed for the first commercial release. Quick buttons remain a simple entry point into scenarios, but they are not the only launch method. The same scenarios must be launched from a regular text request by the user in chat.

Launch Method	Target Backend Logic	Comment
Button on the Assistant screen	The frontend passes a known scenario_type. The backend starts the common scenario contour: clarification -> draft -> preview -> confirm -> save.	The button must not create a separate backend branch.
Text request in chat	The backend / Assistant determines intent_type, asks clarifying questions and returns structured actions for the frontend.	Text requests must lead to the same actions as buttons.
Attachment: photo / material	The backend links the attachment to the message, uses it in the scenario and, if necessary, creates a draft result.	Relevant for calories and Emotion Try-On.

5.2. Quick Buttons on the Assistant Screen

- create an event;
- launch the calorie scenario;
- launch Emotion Try-On / photo-video studio within the scope available for the first release;
- continue a regular chat with the Assistant.

Pressing a button must pass the known scenario type into the unified Assistant scenario contour. The backend response must be compatible with the same draft / preview / confirm mechanism that is used when launching from text.

5.3. Launch from Regular Chat

User Request	Intent	Expected Action
Tomorrow I have a difficult meeting and I am worried I ate an omelet, bread and coffee	event_creation + possible emotion_try_on calorie_tracking	Create an event draft, offer Emotion Try-On and show a preview. Launch the calorie scenario, clarify the portion and show a result preview.
I want to try on calm confidence before the conversation	emotion_try_on	Link Emotion Try-On to an event or offer to create an event.
What is happening to me today?	general_assistant / health_question	Give a structured response without mandatory EventCard creation if there is no action.

5.4. Structured Response Contract

The backend must support the Assistant response not only as text, but also as structured blocks for the frontend.

- quick_replies - quick response options for the user;
- action_buttons - actions: save, edit, cancel, clarify;
- event_card_preview - Event Card preview;
- calorie_result_preview - preview of the calorie scenario result;
- emotion_try_on_preview - preview of the Emotion Try-On result;
- status_block - scenario status: draft_created, waiting_confirmation, saved, cancelled, failed.

Field	Purpose
message	Text response from the Assistant to the user.
blocks	Array of structured blocks for the frontend.
metadata scenario_type	Scenario type: event_creation, calorie_tracking, emotion_try_on,

metadata.source	general_assistant.
metadata.status	Launch source: assistant_button, assistant_chat_text, attachment.
draft_id / pending_action_id	Current scenario status.
	Draft / pending action identifier

5.5. Draft / PendingAction

The Assistant must not save structured objects without user confirmation. For this purpose, a scenario draft / pending action is introduced.

1. The user presses a button or writes a text request.
2. The backend determines the scenario and missing data.
3. The Assistant asks clarifying questions or forms a preliminary result.
4. The backend creates a draft / pending_action.
5. The frontend shows a preview.
6. The user confirms, edits or cancels.
7. The backend saves the EventCard and related data only after confirmation.

Status	Description
draft_created	Draft created, but not yet shown to the user as a final result.
waiting_clarification	The backend is waiting for clarification from the user.
preview_shown	The frontend has shown the preview to the user.
confirmed	The user has confirmed saving.
saved	The backend has successfully saved the structured object.
cancelled	The user has cancelled the action.
failed	The scenario ended with an error; the structured object must not be partially saved without a status.

6. Event Card

6.1. Concept

The Event Card is the core cross-cutting MoodBoss entity. Assistant scenarios, Emotion Try-On and the calorie scenario must not exist separately from the Event Card if their result is meaningful as a fact of the user's day.

6.2. Creation Sources

- calendar;
- quick button on the Assistant screen;
- text request in Assistant chat;
- Emotion Try-On;
- calorie scenario;
- Health contour, if the event is related to the user's state or activity

6.3. Minimum Event Card Composition

Field	Purpose	Comment
event_card_id	Event Card identifier	Required for linkage with Calendar, Health and future SUK.
user_id / profile_id	User and profile	Needed for personal context.
title	Event title	May be proposed by the Assistant and changed by the user.
description	Description	Brief summary of the event or scenario result.
local_date / time	Date and time	Considering the user timezone.
source	Creation source	assistant_button, assistant_chat_text, calendar, health, emotion_try_on.
scenario_type	Scenario type	event_creation, calorie_tracking, emotion_try_on
status	Status	draft, preview_shown, confirmed, saved, cancelled, archived.
assistant_chat_id	Link to topic / chat	If the event was created from the Assistant.
assistant_message_id	Link to message	For result tracing.
health_entry_id	Link to Health	If the event is related to health or calories.
calorie_entry_id	Link to calories	If the event is a meal.

emotion_try_on_result_id

Link to try-on

If an Emotion Try-On result is saved.

6.4. Main Operations

- create an Event Card draft;
- receive an Event Card preview;
- edit the draft before saving;
- confirm saving;
- retrieve the full card by identifier;
- basic update of a saved card;
- archive / delete the card by a separate user action

6.5. Creation Rule After Confirmation

The event_card_created event as the fact of creating a saved card must be recorded only after user confirmation. Before confirmation, draft_id / pending_action_id or an EventCard with draft status is used, if required by the backend model. For SUK and acceptance, only an object with confirmed / saved status is considered a saved card.

7. Calorie Scenario

7.1. General Concept

The calorie scenario is a scenario inside the Assistant. The result must be saved to the Event Card and displayed in the Health section. Manual calorie input as a separate first-level form is not included in the scope of the first release.

7.2. Launch Methods

- pressing the Calories button on the Assistant screen;
- text description of food or drink in chat;
- photo of a dish or drink;
- photo with a text caption;
- clarifying dialogue about portion, composition or meal time.

7.3. Target Scenario

1. The user launches the scenario using a button or text request.
2. The Assistant determines calorie_tracking or receives scenario_type from the button.
3. If data is insufficient, the Assistant asks clarifying questions.
4. The backend calculates incoming calorie value.
5. The frontend receives calorie_result_preview.
6. The user confirms, clarifies or cancels the result.
7. After confirmation, an EventCard of type meal / food and a linked Health / calories record are created.
8. The Health section receives updated balance data.

7.4. Scope of the First Release for Calories

- AI scenario for calorie calculation from photo, text and their combination;
- dialogue-based clarification of portion / composition;
- preview of the result before saving;
- saving incoming calories after confirmation;
- use of existing activity data as the source of expenditure;
- display of current balance at the time of request;
- minimum required API for the calorie balance screen inside Health.

7.5. Not Included in the First Release

- external food database;
- barcode scanning;
- complex nutrient analytics;
- extended diet recommendations;
- separate first-level section for calories;
- manual calorie input as an independent first-level form.

8. Emotion Try-On

8.1. General Concept

Emotion Try-On is a scenario for working with the Event Card inside the Assistant. It must not live separately from the Event Card if the scenario result is saved as a fact or preparation for an event.

8.2. Launch Methods

- button on the Assistant screen;
- user text request in chat;
- Assistant suggestion inside a dialogue about an already created event;
- transition from the Event Card preview.

8.3. Target Scenario

1. The user creates or selects an event.
2. The Assistant offers to launch Emotion Try-On if relevant.
3. The user chooses an emotional mode or formulates it in text.
4. The backend generates the try-on result.
5. The frontend shows emotion_try_on_preview.
6. The user confirms saving the result.
7. The result is saved in the EventCard and linked context of the user's day.

8.4. Scope of the First Release for Emotion Try-On

- scenario launch inside the Assistant;
- receiving a basic result;
- result preview;
- saving the result as part of the EventCard;
- linking the result with the MoodBoss calendar contour.

8.5. Not Included in the First Release

- separate standalone first-level product section;
- extended set of non-standard modes;
- complex behavioral analytics for the try-on scenario;
- complex generative studio beyond the first-release scope, unless described in a separate Technical Specification.

9. Assistant Topics

At the first stage, a topic is treated as a separate chat / case inside the Assistant. Top-level projects combining several chats are not included in the first stage.

- create a topic;
- continue a topic;
- rename a topic;
- delete a topic;
- view the list of topics.

9.1. Data Preservation Rule

Deleting a topic must not automatically delete saved Event Cards, Health data, calorie records, Emotion Try-On results or the long-term user profile. The recommended backend model for the first release is soft delete / archiving of the topic instead of physical deletion if the topic is linked to saved product objects.

10. Backend Contracts and OpenAPI

As part of implementation, OpenAPI and backend contracts must be updated for the following groups of endpoints / operations:

- create / list / rename / delete topics as separate chats / cases;
- send a message to the Assistant with structured response support;
- launch a scenario from a button;
- create draft / pending action;
- receive preview;

- edit draft;
- confirm / cancel draft;
- create and retrieve EventCard;
- save calorie result to EventCard and Health;
- save Emotion Try-On result to EventCard;
- retrieve data for the calorie balance screen inside Health

API Group	Expected Operations
Assistant messages	send message, attach material, return structured blocks, return metadata.
Scenario router	start by button, start by intent, continue scenario, fail gracefully.
Draft / PendingAction	create, get, update, confirm, cancel, expire
EventCard	create draft, preview, confirm, get, update, archive.
Calories	start, clarify, preview, confirm, save, get balance.
Emotion try-on	start, preview, confirm, save result.
Topics	list, create, continue, rename, soft delete.

11. Detailed Data Model

This section is part of the main Technical Specification, not an appendix. It defines the minimum target data model for implementing TS13. Specific table, serializer and class names may be adapted to the current MoodBoss backend architecture, but fields, relationships and statuses must be preserved by meaning.

11.1. EventCard

EventCard is the core cross-cutting MoodBoss entity linking the Assistant, Calendar, Health, calorie scenario and Emotion Try-On.

Field	Type / Format	Mandatory	Description
event_card_id	uuid / string	required	Unique Event Card identifier.
user_id	uuid / string	required	User who owns the card.
profile_id	uuid / string	required	User profile to which the event belongs.
title	string	required after preview	Event title proposed by the Assistant or set by the user.
description	string / nullable	recommended	Description of the event or scenario result.
event_type	enum	required	meeting, meal, emotion_try_on, health, custom, other.
scenario_type	enum	required when created from Assistant	event_creation, calorie_tracking, emotion_try_on, general_assistant.
source	enum	required	assistant_button, assistant_chat_text, calendar, health, emotion_try_on.
status	enum	required	draft, preview_shown, confirmed, saved, cancelled, archived, failed.
local_date	date	required	User local date.
time	time / nullable	optional	Local event time, if applicable.
timezone	string	required	User timezone.
assistant_chat_id / topic_id	uuid / string / nullable	if created from Assistant	Link to Assistant topic.
assistant_message_id	uuid / string / nullable	if created from message	Link to Assistant message.
draft_id / pending_action_id	uuid / string / nullable	before saving	Link to draft.
health_entry_id	uuid / string / nullable	if Health link exists	Link to Health.
calorie_entry_id	uuid / string / nullable	if it is meal / food	Link to calorie record.
emotion_try_on_result_id	uuid / string / nullable	if try-on result exists	Link to Emotion Try-On result.
created_at / updated_at	datetime UTC	required	Service timestamps.
confirmed_at / cancelled_at	datetime UTC / nullable	depending on status	Confirmation or cancellation timestamps.

EventCard relationships: one user may have many EventCards, one EventCard may be linked to one Draft / PendingAction before saving; one EventCard may have one CalorieEntry or one EmotionTryOnResult within the first release; deleting an Assistant topic does not delete the EventCard.

11.2. Draft / PendingAction

Draft / PendingAction records an intermediate scenario result before user confirmation. It is needed for preview, editing, cancel and protection against silent saving.

Field	Type / Format	Mandatory	Description
-------	---------------	-----------	-------------

draft_id / pending_action_id	uuid / string	required	Unique draft identifier.
user_id	uuid / string	required	Draft owner.
profile_id	uuid / string	required	Profile to which the action relates.
assistant_chat_id / topic_id	uuid / string / nullable	if created in chat	Assistant topic.
assistant_message_id	uuid / string / nullable	if created from message	Original user message.
scenario_type	enum	required	event_creation, calorie_tracking, emotion_try_on.
intent_type	enum / nullable	if intent is detected	Recognized user intent.
source	enum	required	assistant_button, assistant_chat_text, attachment
status	enum	required	draft_created, waiting_clarification, preview_shown, confirmed, cancelled, expired, failed.
payload	json	required	Structured draft data.
preview_payload	json / nullable	after preview	Data shown to the user.
idempotency_key	string	required for confirm	Protection key against duplicate saving.
expires_at	datetime UTC / nullable	recommended	Draft lifetime.
created_at / updated_at	datetime UTC	required	Service timestamps.

11.3. CalorieEntry

CalorieEntry records the calorie scenario result after user confirmation. Before confirmation, the result exists only as a preview inside Draft / PendingAction.

Field	Type / Format	Mandatory	Description
calorie_entry_id	uuid / string	required after save	Unique calorie record identifier.
event_card_id	uuid / string	required	Link to EventCard of type meal / food.
health_entry_id	uuid / string / nullable	if Health record is created	Link to Health.
user_id / profile_id	uuid / string	required	Owner and profile.
input_type	enum	required	text, photo, text_photo.
source_text	string / nullable	if text existed	User description of food.
attachment_id	uuid / string / nullable	if photo existed	Link to uploaded material.
meal_type	enum / nullable	recommended	breakfast, lunch, dinner, snack, drink, unknown.
estimated_kcal	number	required after calculation	Estimated incoming calories.
confidence	number / nullable	recommended	Conditional calculation confidence.
clarifications	json / nullable	if clarifications existed	Portion, composition, meal time.
status	enum	required	preview_shown, confirmed, saved, cancelled, failed.
created_at / confirmed_at	datetime UTC	required after save	Creation and confirmation timestamps.

11.4. EmotionTryOnResult

EmotionTryOnResult records the Emotion Try-On result if the user has confirmed saving the result to the Event Card.

Field	Type / Format	Mandatory	Description
emotion_try_on_result_id	uuid / string	required after save	Unique try-on result identifier.
event_card_id	uuid / string	required	Link to EventCard.
user_id / profile_id	uuid / string	required	Owner and profile.
emotion_mode	string / enum	required	Selected emotional mode, for example calm_confidence.
input_text	string / nullable	if text existed	User wording.
attachment_id	uuid / string / nullable	if material existed	Link to photo / video / material.
result_summary	string	required	Brief result description for the user.
result_payload	json / nullable	if structured result exists	Structured try-on result.
status	enum	required	preview_shown, confirmed, saved, cancelled, failed.
created_at / confirmed_at	datetime UTC	required after save	Creation and confirmation timestamps.

11.5. AssistantTopic

AssistantTopic is a topic / chat / case inside the Assistant. In the first release it is not a top-level project and does not combine several projects.

Field	Type / Format	Mandatory	Description
assistant_chat_id / topic_id	uuid / string	required	Unique topic identifier.
user_id / profile_id	uuid / string	required	Owner and profile.
title	string	required	Topic title set by the user or proposed by the Assistant.
status	enum	required	active, archived, deleted_soft.
last_message_id	uuid / string / nullable	recommended	Last message in the topic.
created_at / updated_at	datetime UTC	required	Service timestamps.
deleted_at	datetime UTC / nullable	if deleted	Soft delete timestamp

Deleting AssistantTopic must be performed as soft delete / archiving if the topic is linked to EventCard, CalorieEntry, EmotionTryOnResult or Health records.

12. JSON Contracts for Backend Responses

This section is part of the main Technical Specification. JSON examples define the target backend response format for the frontend. Fields may be extended, but the basic message / blocks / metadata / status / ids structure must be preserved.

12.1. EventCard preview

```
{
  "message": "I have prepared the Event Card. Please check the data before saving.",
  "blocks": [
    {
      "type": "event_card_preview",
      "draft_id": "draft_123",
      "event_card": {
        "title": "Difficult meeting",
        "description": "The user is worried before the meeting",
        "local_date": "2026-05-01",
        "time": "10:00",
        "event_type": "meeting",
        "status": "preview_shown"
      }
    },
    {
      "type": "actions",
      "actions": [
        "confirm",
        "edit",
        "cancel"
      ]
    }
  ],
  "metadata": {
    "scenario_type": "event_creation",
    "source": "assistant_chat_text",
    "intent_type": "event_creation",
    "status": "preview_shown"
  }
}
```

12.2. Calorie preview

```
{
  "message": "I have estimated the calories. Please confirm or clarify the portion.",
  "blocks": [
    {
      "type": "calorie_result_preview",
      "draft_id": "draft_456",

```



```
"calorie_result": {
  "input_type": "text_photo",
  "estimated_kcal": 420,
  "meal_type": "breakfast",
  "items": [
    {
      "name": "omelet",
      "estimated_kcal": 250
    },
    {
      "name": "bread",
      "estimated_kcal": 120
    },
    {
      "name": "coffee",
      "estimated_kcal": 50
    }
  ],
  "status": "preview_shown"
},
"actions": [
  "confirm",
  "clarify",
  "cancel"
]
}
},
"metadata": {
  "scenario_type": "calorie_tracking",
  "source": "assistant_chat_text",
  "status": "preview_shown"
}
}
```

12.3. Emotion try-on preview

```
{
  "message": "I have prepared the Emotion Try-On result. It can be saved to the Event Card.",
  "blocks": [
    {
      "type": "emotion_try_on_preview",
      "draft_id": "draft_789",
      "emotion_try_on": {
        "emotion_mode": "calm_confidence",
        "result_summary": "Calm confidence before the conversation",
        "status": "preview_shown"
      },
      "actions": [
        "confirm",
        "edit",
        "cancel"
      ]
    }
  ],
  "metadata": {
    "scenario_type": "emotion_try_on",
    "source": "assistant_chat_text",
  }
}
```

```
"status": "preview_shown"
}
}
```

12.4. Confirm response

```
{
  "message": "Done. The result has been saved.",
  "blocks": [
    {
      "type": "status_block",
      "status": "saved"
    },
    {
      "type": "event_card_preview",
      "event_card_id": "event_123",
      "actions": [
        "open",
        "edit",
        "archive"
      ]
    }
  ],
  "metadata": {
    "draft_id": "draft_123",
    "event_card_id": "event_123",
    "calorie_entry_id": null,
    "emotion_try_on_result_id": null,
    "status": "saved",
    "idempotency_key": "confirm_abc"
  }
}
```

12.5. Cancel response

```
{
  "message": "The action has been cancelled. I have not saved anything.",
  "blocks": [
    {
      "type": "status_block",
      "status": "cancelled"
    }
  ],
  "metadata": {
    "draft_id": "draft_123",
    "status": "cancelled"
  }
}
```

13. API Errors and Statuses

The backend must return errors in a unified format so that the frontend can show a clear status_block and does not create partially saved objects.

HTTP code	error_code	When It Occurs	Expected Behavior
400	invalid_payload	Incorrect request body, missing required fields, invalid enum format	The frontend shows an input error and does not move the scenario to saved.
403	profile_mismatch	The submitted draft_id, event_card_id or profile_id does not belong to the current user / profile	The operation is blocked and the object is not changed

404	draft_not_found	The draft was not found or was deleted.	The frontend offers to start the scenario again.
409	draft_expired	The draft / pending action has expired.	The frontend shows expired status and offers to create a new draft.
409	already_confirmed	Repeated confirm for an already confirmed draft.	The backend returns the previously created ids without creating duplicates.
422	missing_clarification	Calculation or EventCard lacks data: date, portion, emotional mode, etc.	The backend returns quick_replies / a clarifying question.
500	scenario_failed	Internal scenario error or external AI service error.	The backend does not create a partially saved object without failed status.

13.1. Unified Error Format

```
{
  "error": {
    "code": "missing_clarification",
    "message": "The dish portion needs to be clarified.",
    "status": "waiting_clarification",
    "details": {
      "missing_fields": [
        "portion_size"
      ]
    }
  },
  "draft_id": "draft_456"
},
"blocks": [
  {
    "type": "quick_replies",
    "items": [
      "small portion",
      "medium portion",
      "large portion"
    ]
  },
  {
    "type": "status_block",
    "status": "waiting_clarification"
  }
]
}
```

13.2. Scenario Statuses

Status	Meaning	Can the Object Be Saved
draft_created	Draft created, the user has not yet confirmed.	No.
waiting_clarification	The backend is waiting for clarifying data.	No.
preview_shown	Preview shown to the user.	No, until confirm.
confirmed	The user has confirmed the action.	Yes, the backend may perform save.
saved	The object has been saved.	Already saved.
cancelled	The user has cancelled the action.	No.
expired	The draft is outdated.	No.
failed	The scenario ended with an error.	No without separate recovery.

14. Idempotency and Duplicate Protection Rules

Repeated confirm must not create two EventCards, two CalorieEntries or two EmotionTryOnResults. The backend must ensure idempotency of confirmation operations.

14.1. General Rules

- Each confirm operation must accept or form an idempotency_key.
- For one draft_id, only one successful transition to confirmed / saved is allowed
- If the frontend sends confirm again with the same idempotency_key, the backend returns the previously created result
- If confirm is sent again with another idempotency_key while the draft is already saved, the backend does not create a duplicate and returns already_confirmed / previously created ids.
- Creation of EventCard, CalorieEntry and EmotionTryOnResult must be performed in a transaction or equivalent atomic block.
- In case of failure after partial saving, the backend must be able to return a consistent status or perform rollback.

14.2. Minimum Confirm Logic

1. Check the user and profile.
2. Find draft / pending_action.
3. Check draft status draft_created, preview_shown or waiting_clarification may continue, cancelled / expired / failed are not saved.
4. Check idempotency_key.
5. If the result has already been saved, return existing event_card_id / calorie_entry_id / emotion_try_on_result_id.
6. If the result does not yet exist, save it in a transaction.
7. Update statuses of the draft and related objects.
8. Return confirm response with final identifiers.

14.3. Unique Constraints

Object	Recommended Constraint	Why It Is Needed
Draft / PendingAction	unique(draft_id, user_id, profile_id)	Exclude access to another user's draft.
Confirm operation	unique(draft_id, idempotency_key)	Exclude repeated processing of one confirm.
EventCard from draft	unique(draft_id) where status in confirmed/saved	Do not create two cards from one draft.
CalorieEntry from draft	unique(draft_id) for calorie_tracking	Do not create two calorie records from one draft.
EmotionTryOnResult from draft	unique(draft_id) for emotion_try_on	Do not create two try-on results from one draft.

15. Technical Readiness for SUK

A full SUK for these events remains under the separate Technical Specification SE-01-TS11. Within TS13, the backend must ensure technical readiness for subsequent integration: object identifiers, creation sources, scenario statuses, timestamps and links between entities.

15.1. Fields That Must Be Available Within TS13

- user_id;
- profile_id;
- assistant_chat_id / topic_id;
- assistant_message_id;
- scenario_type;
- scenario_source;
- intent_type;
- draft_id;
- event_card_id;
- health_entry_id;
- calorie_entry_id;
- emotion_try_on_result_id;
- source_screen;
- source_action;
- status;

- created_at / updated_at / confirmed_at / cancelled_at;
- local_date;
- timezone;
- language;
- platform;
- app_version.

15.2. Events for the Subsequent SUK Technical Specification

- assistant_scenario_started;
- assistant_intent_detected;
- assistant_question_asked;
- assistant_action_shown;
- assistant_action_clicked;
- assistant_draft_created;
- assistant_preview_shown;
- assistant_draft_confirmed;
- assistant_draft_cancelled;
- event_card_created;
- event_card_created_from_assistant;
- calorie_scenario_started;
- calorie_result_preview_shown;
- calorie_result_saved;
- emotion_try_on_started;
- emotion_try_on_preview_shown;
- emotion_try_on_result_saved;
- assistant_scenario_completed;
- assistant_scenario_abandoned.

15.3. Not Included in TS13 Regarding SUK

- SUK showcases;
- management reports;
- aggregates and recurring calculations;
- analytics dashboards;
- extended product metrics matrix;
- budget and advertising analytics for these events.

16. Security and Data Preservation Requirements

- The Assistant must not save EventCard, calorie result or Emotion Try-On result without user confirmation;
- deleting a topic / chat must not delete saved structured product objects;
- user_id / profile_id checks must be applied to save, edit and delete operations;
- attachments must be linked to a specific message and scenario;
- when a scenario error occurs, the user must receive a clear status_block, and the backend must not create partially saved objects without an explicit status;
- all identifiers returned by the frontend must be checked for ownership by the user and profile.

17. Critical End-to-End Acceptance Scenarios

Scenario	Verification Steps	Success Criterion
Event from button	Assistant -> Create Event button -> clarifications -> draft -> preview -> confirm -> EventCard -> Calendar	The card is saved and displayed in the calendar contour.
Event from text	User writes Tomorrow difficult meeting -> intent -> clarifications -> preview -> confirm	EventCard created from chat, linked to topic and message.
Calories from chat	User writes food description or attaches a photo ->	EventCard of type meal is created;

Emotion Try-On	calculation -> preview -> confirm User creates / selects an event -> try-on -> preview -> confirm	data is displayed in Health. The try-on result is saved in EventCard.
Topics	Create topic, continue, rename, delete	The topic is managed as chat / case; saved EventCard and Health are not deleted.
Technical readiness for SUK	Check presence of ids, source, status, timestamps, links	The backend returns required fields for the future SUK Technical Specification.
Confirm idempotency	Send confirm twice for one draft_id / idempotency_key	The second operation returns previously created ids and does not create a duplicate

18. Scope Limitations of the First Release

- top-level projects combining several topics / chats;
- full SUK, showcases, reports and aggregates;
- external food database and barcode scanning;
- complex recurring events and extended event lifecycle;
- advanced nutrient analytics and dietary recommendations;
- separate first-level product section for Emotion Try-On or calories;
- separate frontend/mobile work, except for contract alignment and integration support.

19. Work Result

- hybrid Assistant model with scenario launch from buttons and chat;
- user intent understanding for key scenarios;
- clarifying questions and structured actions for the frontend;
- draft / preview / confirm flow;
- EventCard creation and saving;
- saving calorie result to EventCard and Health;
- saving Emotion Try-On result to EventCard;
- topics as separate chats / cases;
- detailed data model for EventCard, Draft / PendingAction, CalorieEntry, EmotionTryOnResult and AssistantTopic;
- JSON contracts for backend responses for preview, confirm and cancel;
- unified API error and status format;
- confirm idempotency and duplicate protection;
- technical readiness for the subsequent SUK Technical Specification

Signatures of the Parties

Contractor
Vistadi LLC
Director: _____
Seal _____


Dildin V.S.

Customer
ONERY OVERSEAS LIMITED
Director:  Boulitsidou A.
Seal _____