# Journal Buddy

**Favour Akpasi (k12248660), Kilian Ludwig (k11922175)**

# Table of contents

1. Goal of Journal Buddy
2. Requirements
3. Use case descriptions
4. Use case diagram
5. Traceability matrix
6. Domain model
7. Architecture diagram
8. Components description
9. Design questions and answers

# Goal of Journal Buddy

- **Conversational Journal Interface:** Utilises Large Language Models to provide a virtual friend experience, enabling users to interact with their journal entries

- **Enhanced Engagement:** Makes journaling more engaging and interactive by facilitating discussions and reflections on entries

- **AI-Powered Insights:** Uses AI to analyse journal content, offering personalised suggestions and related questions

# Requirements

## Functional requirements

1 - The Text processing **shall** be performed in less than 5s.

2 - The JournalFriend **shall** interact with users via text prompts.

3 - **If** the User Requests conversation about an entry or Reflection, the JournalFriend **shall** provide the response using a typewriter or text effect to make the conversation feel natural.

4 - **While** processing response, the JournalFriend **shall** display a typing indicator to let a user know that a response is being generated.

5 - The JournalFriend **shall** store the Users inputs and its Response in a Secured Database only accessible via Authentication

6 - The Language of operation of the JournalFriend **shall** be English.

## Non-functional requirements

1-The JornalFriend **shall** allow users to authenticate with their credentials

2 - **When** the user authenticates, the JornalFriend **shall** display the home screen with a text box and previous chats if available

3 - **When** the User inputs a Journal (Text) entry, the JournalFriend **shall** Store the Text in a Database

4 - **When** the User Requests conversation about an entry, the JournalFriend **shall** process the Journal Entry and Provide a response

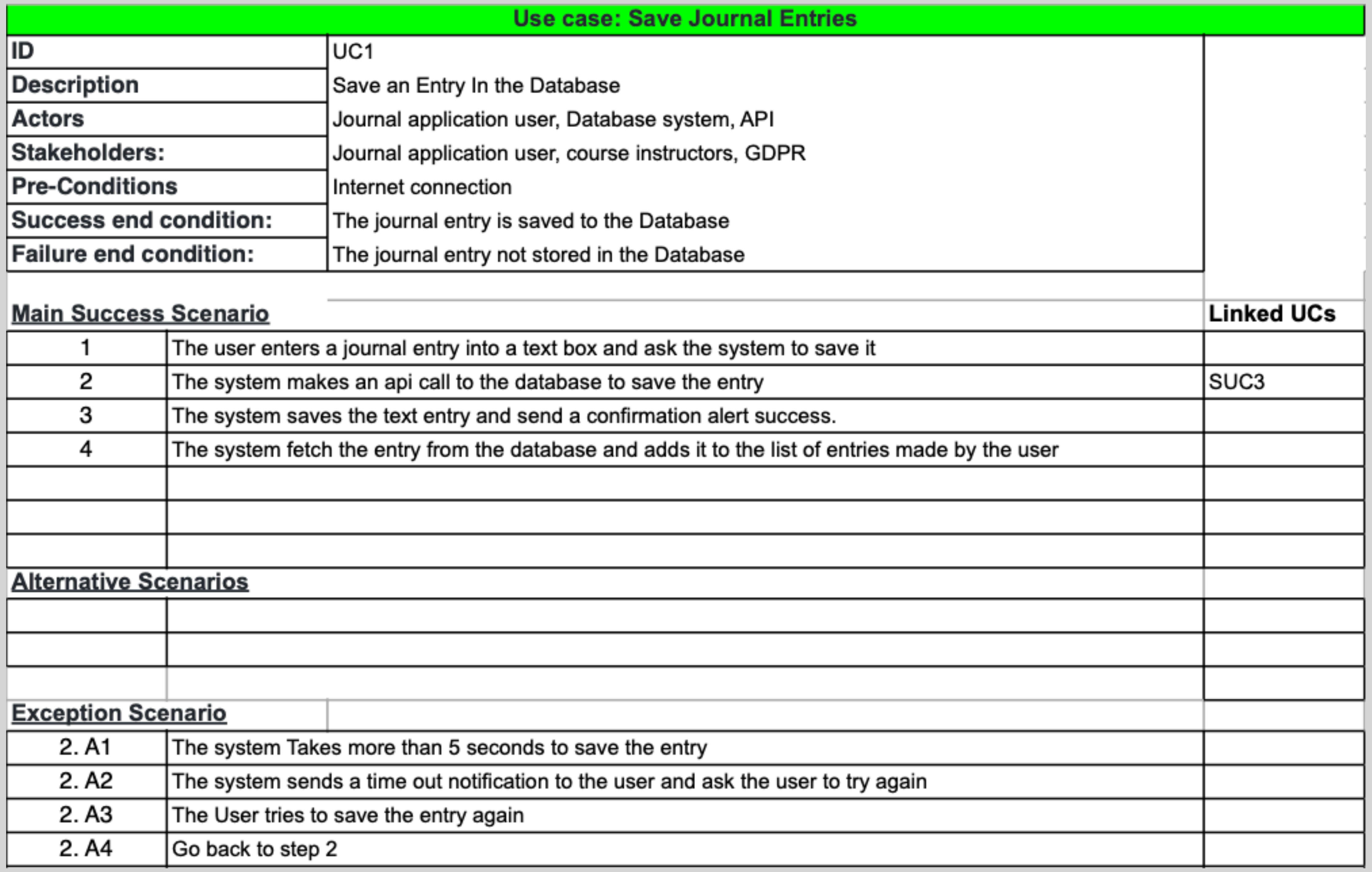
5 - **If** the User Requests Reflection, the JournalFriend **shall** collect analyse and process all the entries specified in the selected Reflection period and provide an appropriate summary as Response

## AI related requirements

The language model shall support sophisticated large parameter checkpoints using the OpenAI API, making the responses meaningful.

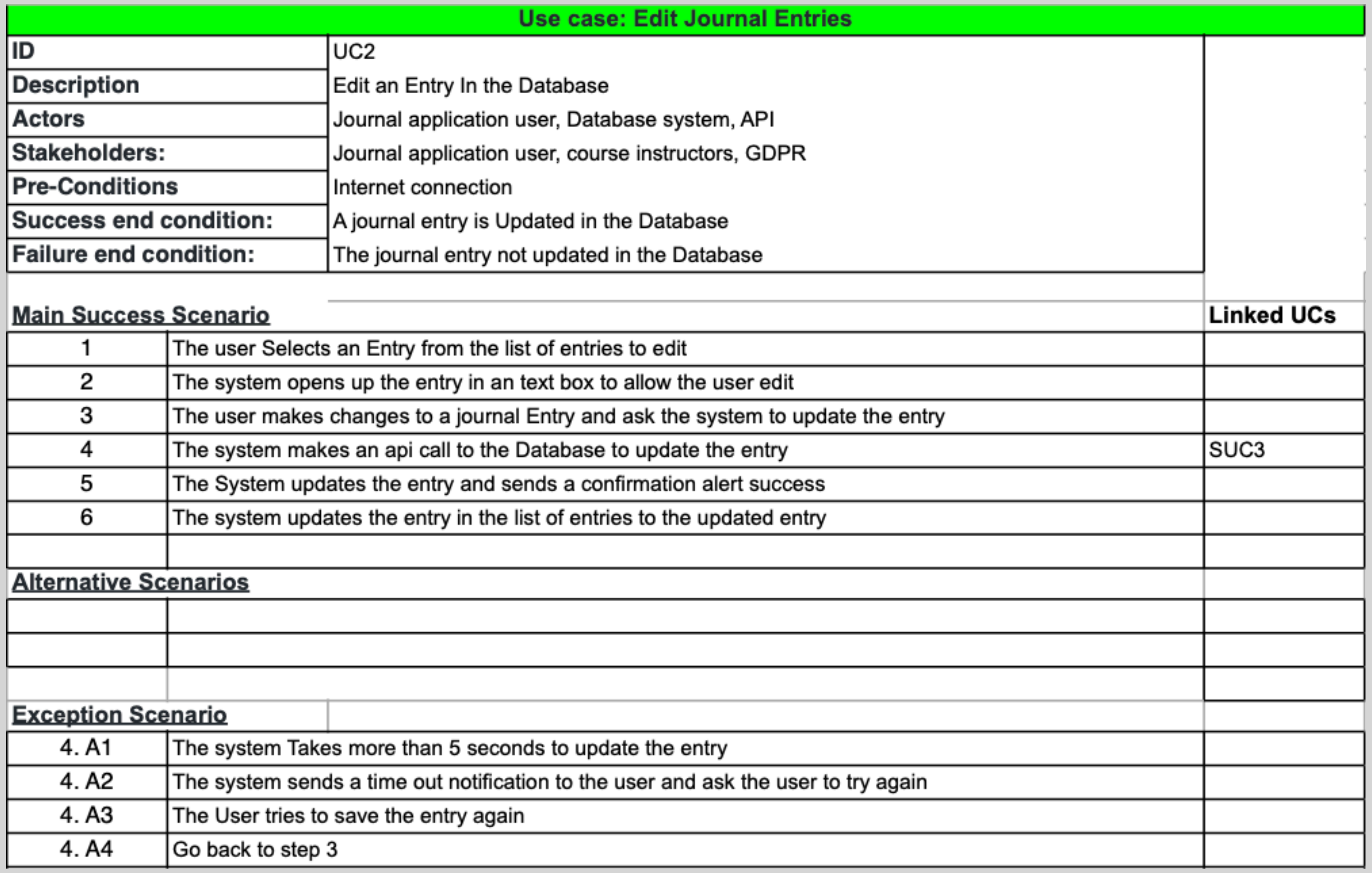
The response generation shall support the stream functionality of the OpenAI API, making the conversation feel natural and responsive.

# Use cases 1 - 2



| Use case: Save Journal Entries | | |
|---|---|---|
| ID | UC1 | |
| Description | Save an Entry In the Database | |
| Actors | Journal application user, Database system, API | |
| Stakeholders: | Journal application user, course instructors, GDPR | |
| Pre-Conditions | Internet connection | |
| Success end condition: | The journal entry is saved to the Database | |
| Failure end condition: | The journal entry not stored in the Database | |
| Main Success Scenario | | Linked UCs |
| 1 | The user enters a journal entry into a text box and ask the system to save it | |
| 2 | The system makes an api call to the database to save the entry | SUC3 |
| 3 | The system saves the text entry and send a confirmation alert success. | |
| 4 | The system fetch the entry from the database and adds it to the list of entries made by the user | |
| | | |
| | | |
| | | |
| Alternative Scenarios | | |
| | | |
| | | |
| | | |
| Exception Scenario | | |
| 2. A1 | The system Takes more than 5 seconds to save the entry | |
| 2. A2 | The system sends a time out notification to the user and ask the user to try again | |
| 2. A3 | The User tries to save the entry again | |
| 2. A4 | Go back to step 2 | |

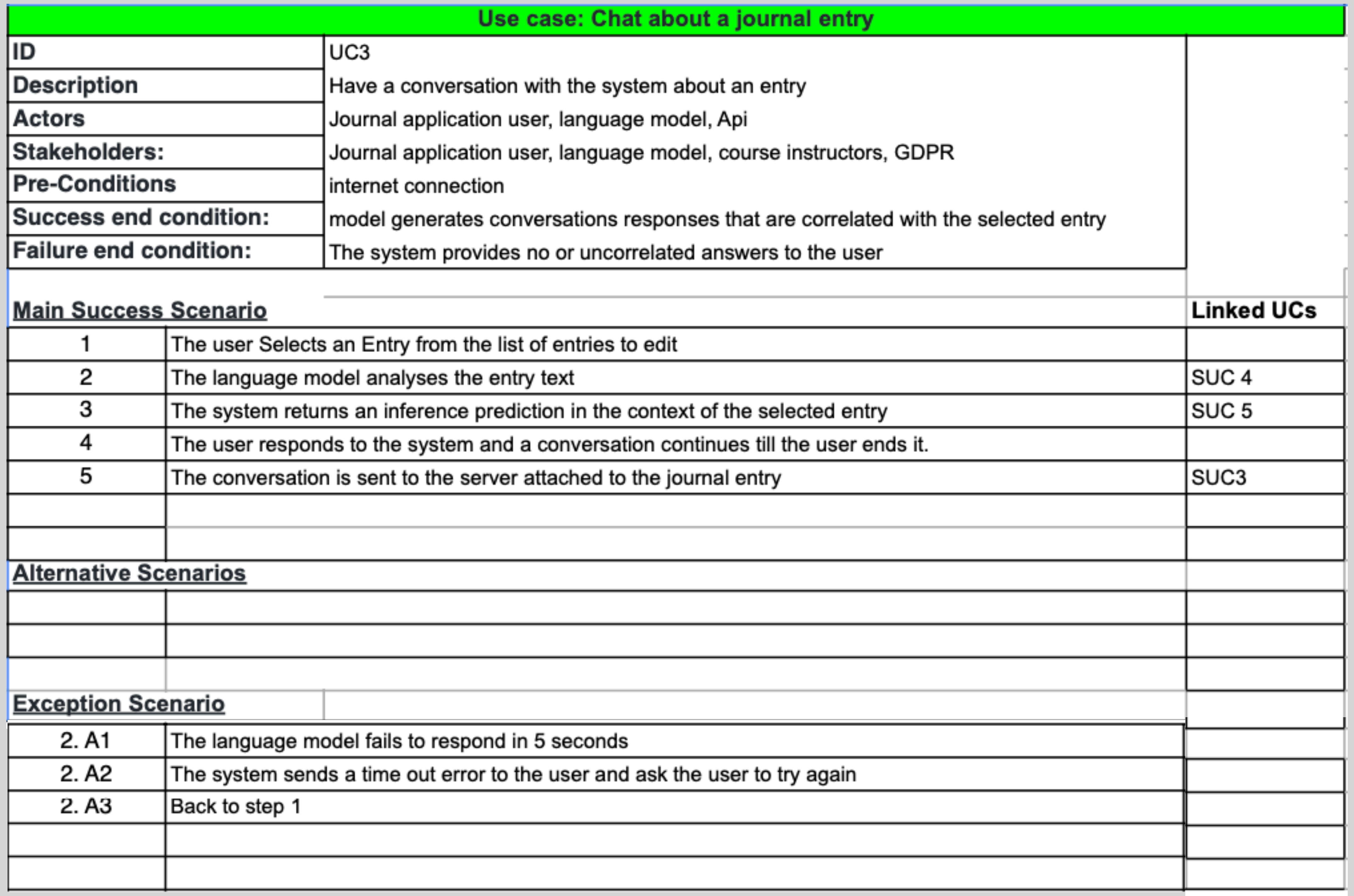
✓ Implemented



| Use case: Edit Journal Entries | | |
|---|---|---|
| ID | UC2 | |
| Description | Edit an Entry In the Database | |
| Actors | Journal application user, Database system, API | |
| Stakeholders: | Journal application user, course instructors, GDPR | |
| Pre-Conditions | Internet connection | |
| Success end condition: | A journal entry is Updated in the Database | |
| Failure end condition: | The journal entry not updated in the Database | |
| Main Success Scenario | | Linked UCs |
| 1 | The user Selects an Entry from the list of entries to edit | |
| 2 | The system opens up the entry in an text box to allow the user edit | |
| 3 | The user makes changes to a journal Entry and ask the system to update the entry | |
| 4 | The system makes an api call to the Database to update the entry | SUC3 |
| 5 | The System updates the entry and sends a confirmation alert success | |
| 6 | The system updates the entry in the list of entries to the updated entry | |
| | | |
| Alternative Scenarios | | |
| | | |
| | | |
| | | |
| Exception Scenario | | |
| 4. A1 | The system Takes more than 5 seconds to update the entry | |
| 4. A2 | The system sends a time out notification to the user and ask the user to try again | |
| 4. A3 | The User tries to save the entry again | |
| 4. A4 | Go back to step 3 | |

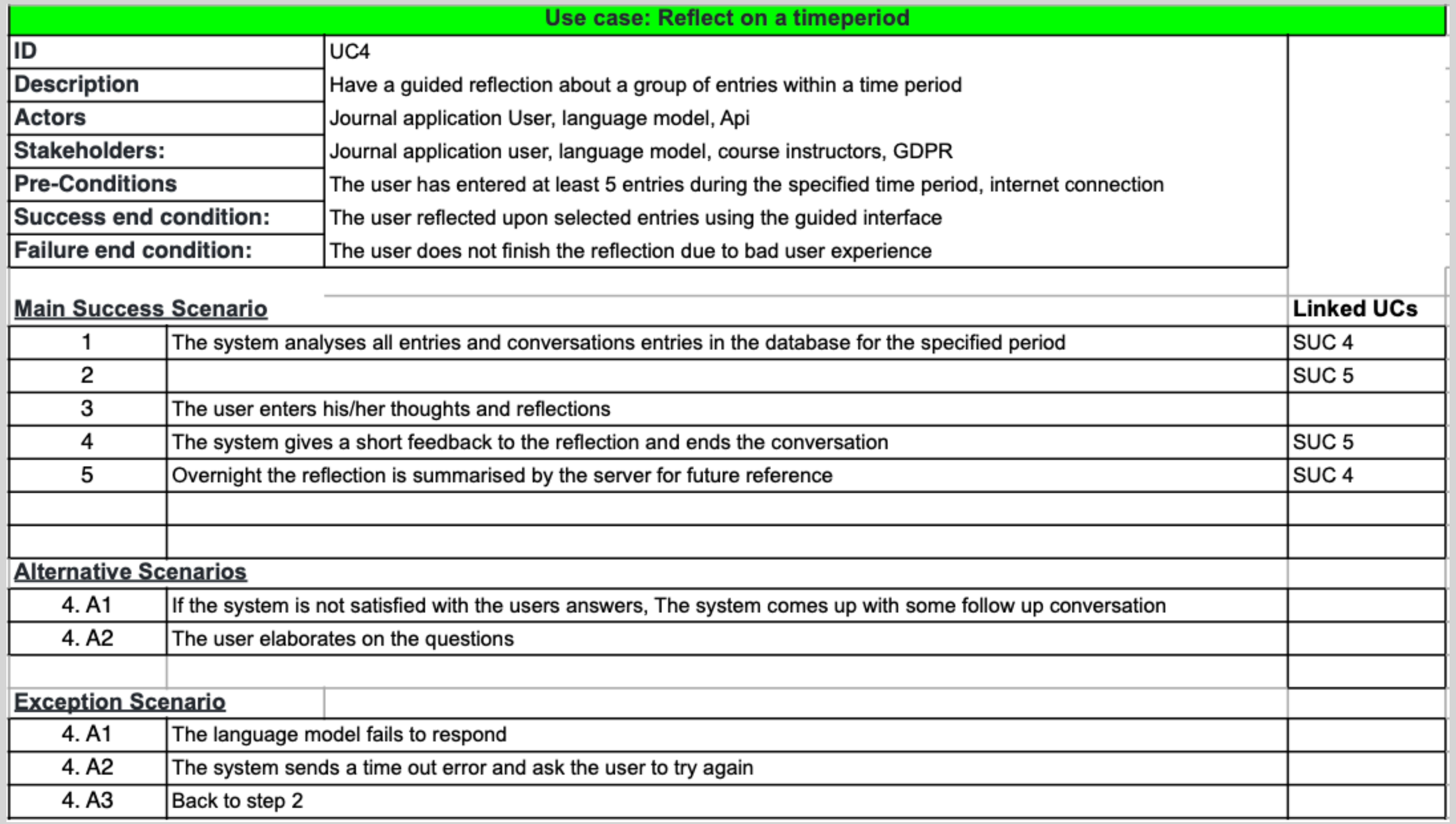
✓ Implemented

# Use cases 3 - 4



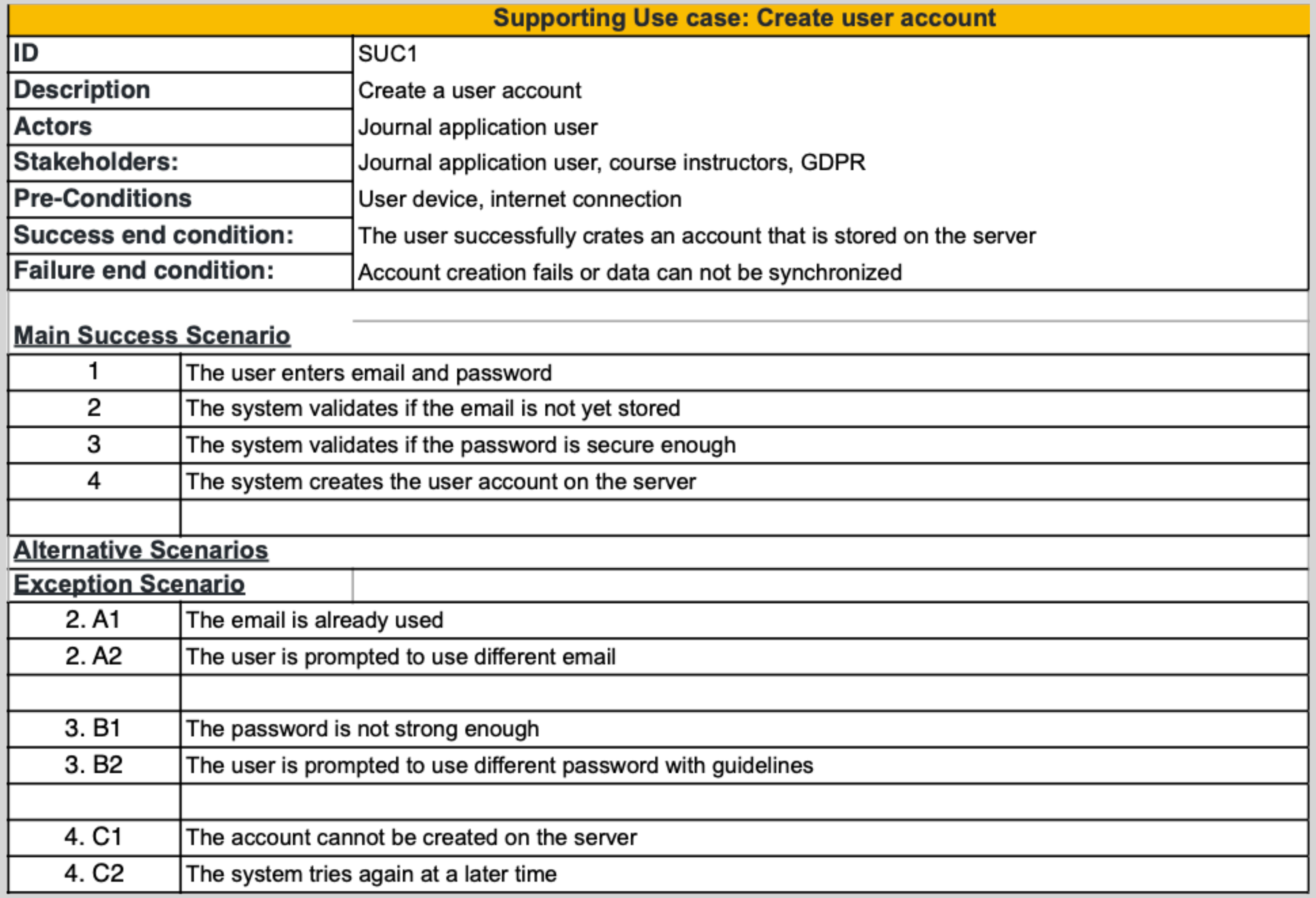
| Use case: Chat about a journal entry | | |
|---|---|---|
| ID | UC3 | |
| Description | Have a conversation with the system about an entry | |
| Actors | Journal application user, language model, Api | |
| Stakeholders: | Journal application user, language model, course instructors, GDPR | |
| Pre-Conditions | internet connection | |
| Success end condition: | model generates conversations responses that are correlated with the selected entry | |
| Failure end condition: | The system provides no or uncorrelated answers to the user | |
| Main Success Scenario | | Linked UCs |
| 1 | The user Selects an Entry from the list of entries to edit | |
| 2 | The language model analyses the entry text | SUC 4 |
| 3 | The system returns an inference prediction in the context of the selected entry | SUC 5 |
| 4 | The user responds to the system and a conversation continues till the user ends it. | |
| 5 | The conversation is sent to the server attached to the journal entry | SUC3 |
| | | |
| | | |
| Alternative Scenarios | | |
| | | |
| | | |
| | | |
| Exception Scenario | | |
| 2. A1 | The language model fails to respond in 5 seconds | |
| 2. A2 | The system sends a time out error to the user and ask the user to try again | |
| 2. A3 | Back to step 1 | |
| | | |
| | | |

✓ Implemented



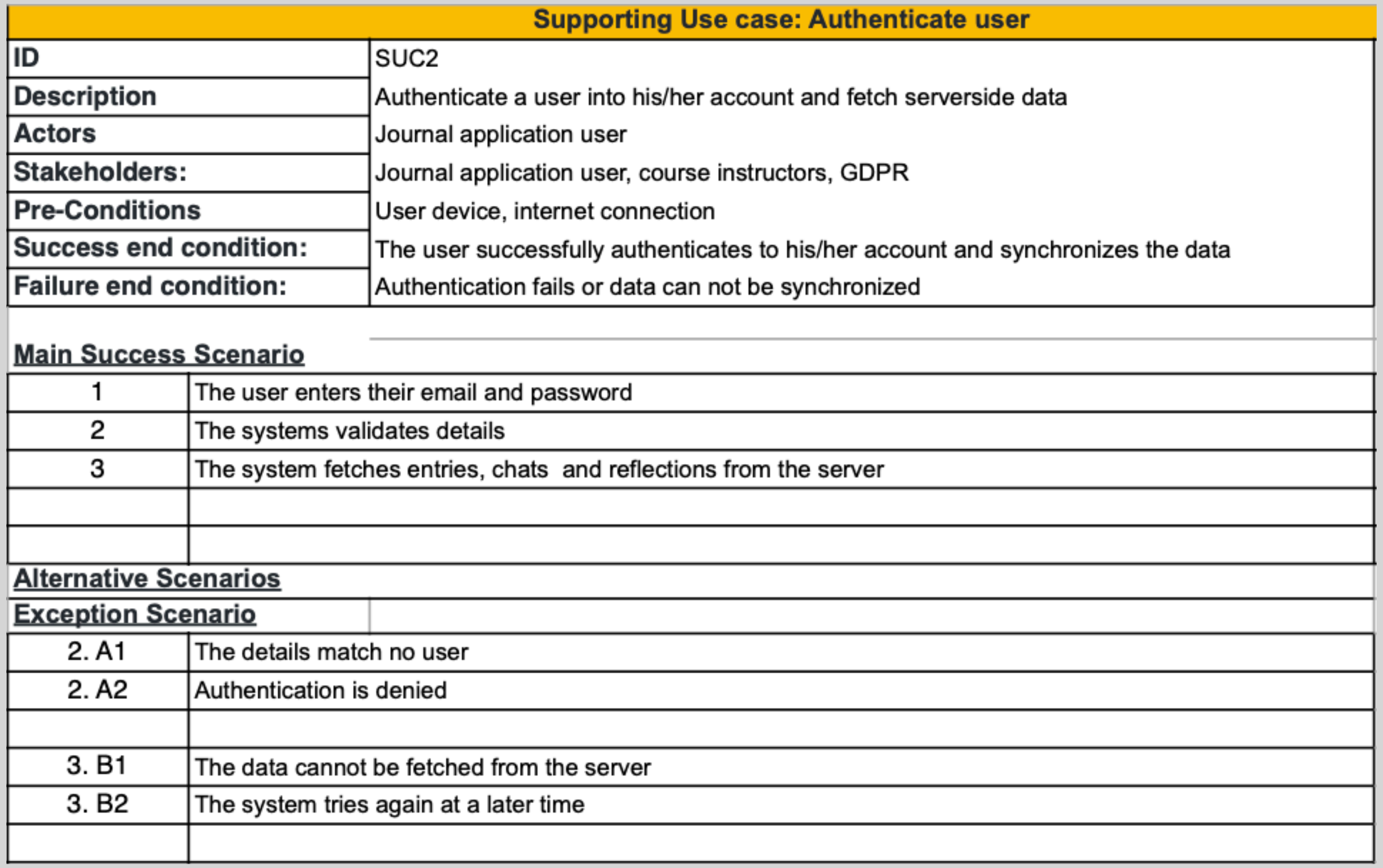
| Use case: Reflect on a timeperiod | | |
|---|---|---|
| ID | UC4 | |
| Description | Have a guided reflection about a group of entries within a time period | |
| Actors | Journal application User, language model, Api | |
| Stakeholders: | Journal application user, language model, course instructors, GDPR | |
| Pre-Conditions | The user has entered at least 5 entries during the specified time period, internet connection | |
| Success end condition: | The user reflected upon selected entries using the guided interface | |
| Failure end condition: | The user does not finish the reflection due to bad user experience | |
| Main Success Scenario | | Linked UCs |
| 1 | The system analyses all entries and conversations entries in the database for the specified period | SUC 4 |
| 2 | | SUC 5 |
| 3 | The user enters his/her thoughts and reflections | |
| 4 | The system gives a short feedback to the reflection and ends the conversation | SUC 5 |
| 5 | Overnight the reflection is summarised by the server for future reference | SUC 4 |
| | | |
| | | |
| Alternative Scenarios | | |
| 4. A1 | If the system is not satisfied with the users answers, The system comes up with some follow up conversation | |
| 4. A2 | The user elaborates on the questions | |
| | | |
| Exception Scenario | | |
| 4. A1 | The language model fails to respond | |
| 4. A2 | The system sends a time out error and ask the user to try again | |
| 4. A3 | Back to step 2 | |

# Supporting use cases 1 - 2



| | |
|---|---|
| **Supporting Use case: Create user account** | |
| **ID** | SUC1 |
| **Description** | Create a user account |
| **Actors** | Journal application user |
| **Stakeholders:** | Journal application user, course instructors, GDPR |
| **Pre-Conditions** | User device, internet connection |
| **Success end condition:** | The user successfully crates an account that is stored on the server |
| **Failure end condition:** | Account creation fails or data can not be synchronized |
| **Main Success Scenario** | |
| 1 | The user enters email and password |
| 2 | The system validates if the email is not yet stored |
| 3 | The system validates if the password is secure enough |
| 4 | The system creates the user account on the server |
| | |
| **Alternative Scenarios** | |
| **Exception Scenario** | |
| 2. A1 | The email is already used |
| 2. A2 | The user is prompted to use different email |
| | |
| 3. B1 | The password is not strong enough |
| 3. B2 | The user is prompted to use different password with guidelines |
| | |
| 4. C1 | The account cannot be created on the server |
| 4. C2 | The system tries again at a later time |

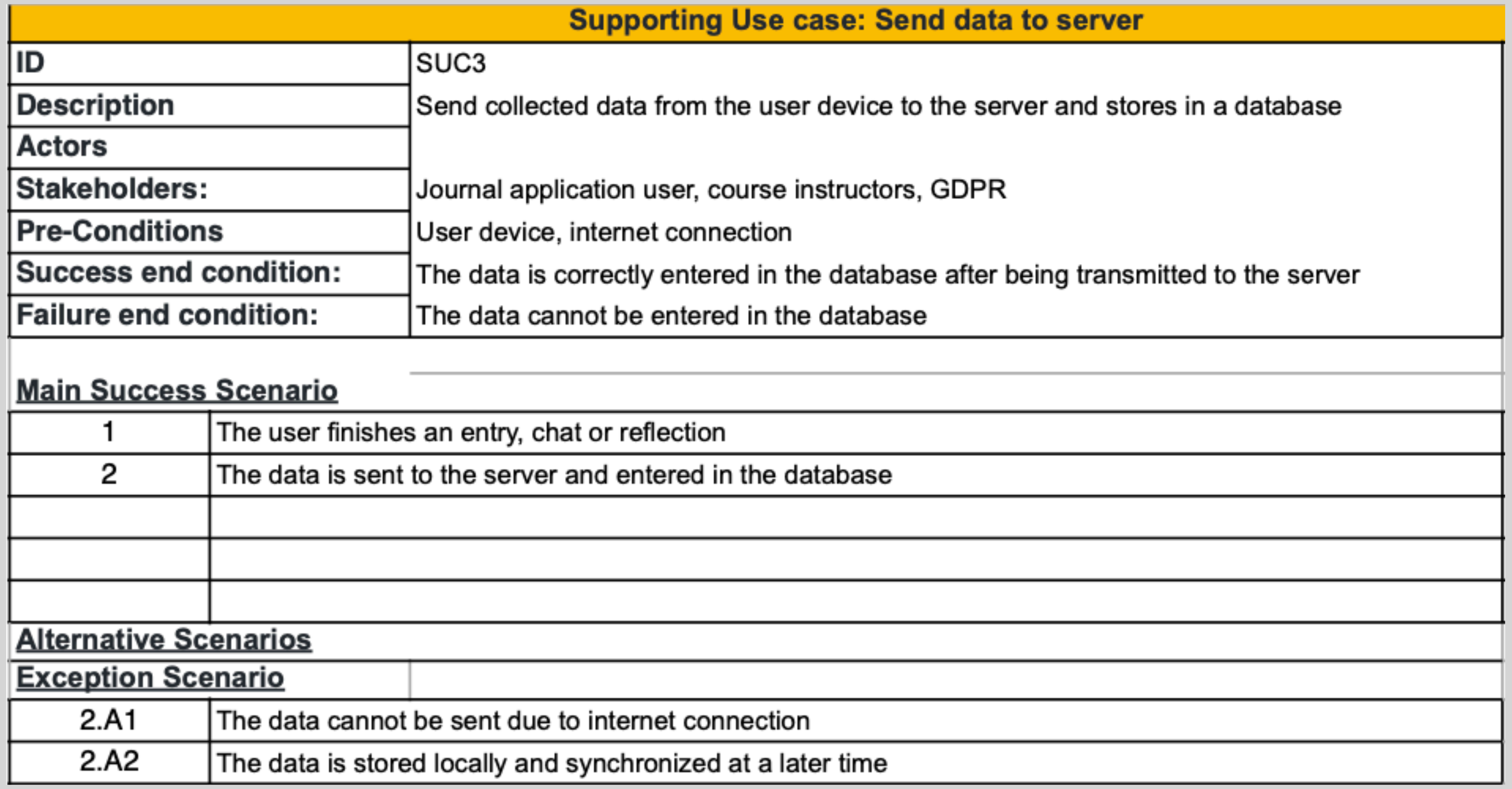
✓ Implemented



| | |
|---|---|
| **Supporting Use case: Authenticate user** | |
| **ID** | SUC2 |
| **Description** | Authenticate a user into his/her account and fetch serverside data |
| **Actors** | Journal application user |
| **Stakeholders:** | Journal application user, course instructors, GDPR |
| **Pre-Conditions** | User device, internet connection |
| **Success end condition:** | The user successfully authenticates to his/her account and synchronizes the data |
| **Failure end condition:** | Authentication fails or data can not be synchronized |
| **Main Success Scenario** | |
| 1 | The user enters their email and password |
| 2 | The systems validates details |
| 3 | The system fetches entries, chats and reflections from the server |
| | |
| | |
| **Alternative Scenarios** | |
| **Exception Scenario** | |
| 2. A1 | The details match no user |
| 2. A2 | Authentication is denied |
| | |
| 3. B1 | The data cannot be fetched from the server |
| 3. B2 | The system tries again at a later time |
| | |

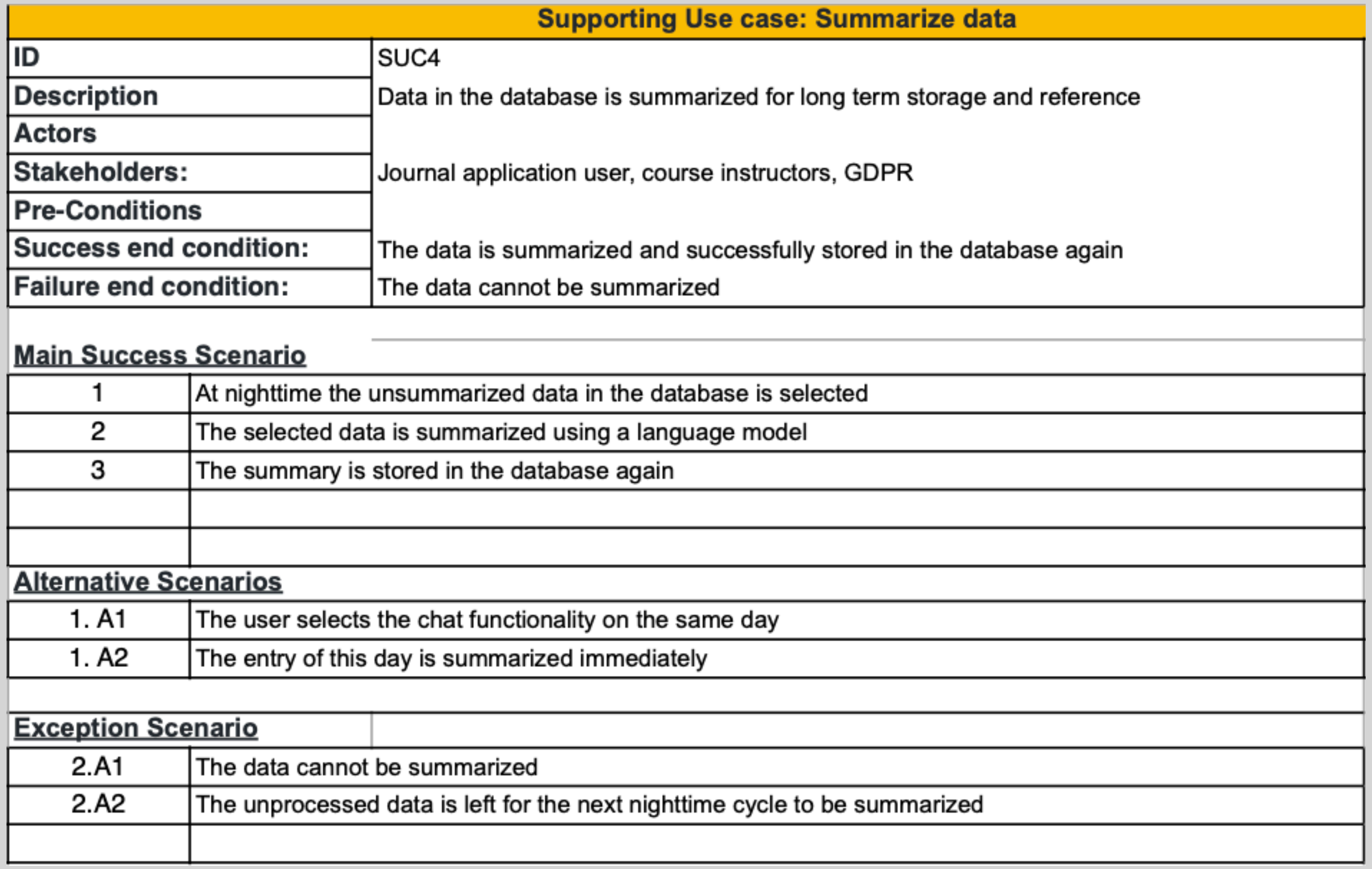
✓ Implemented

# Supporting use cases 3 - 4



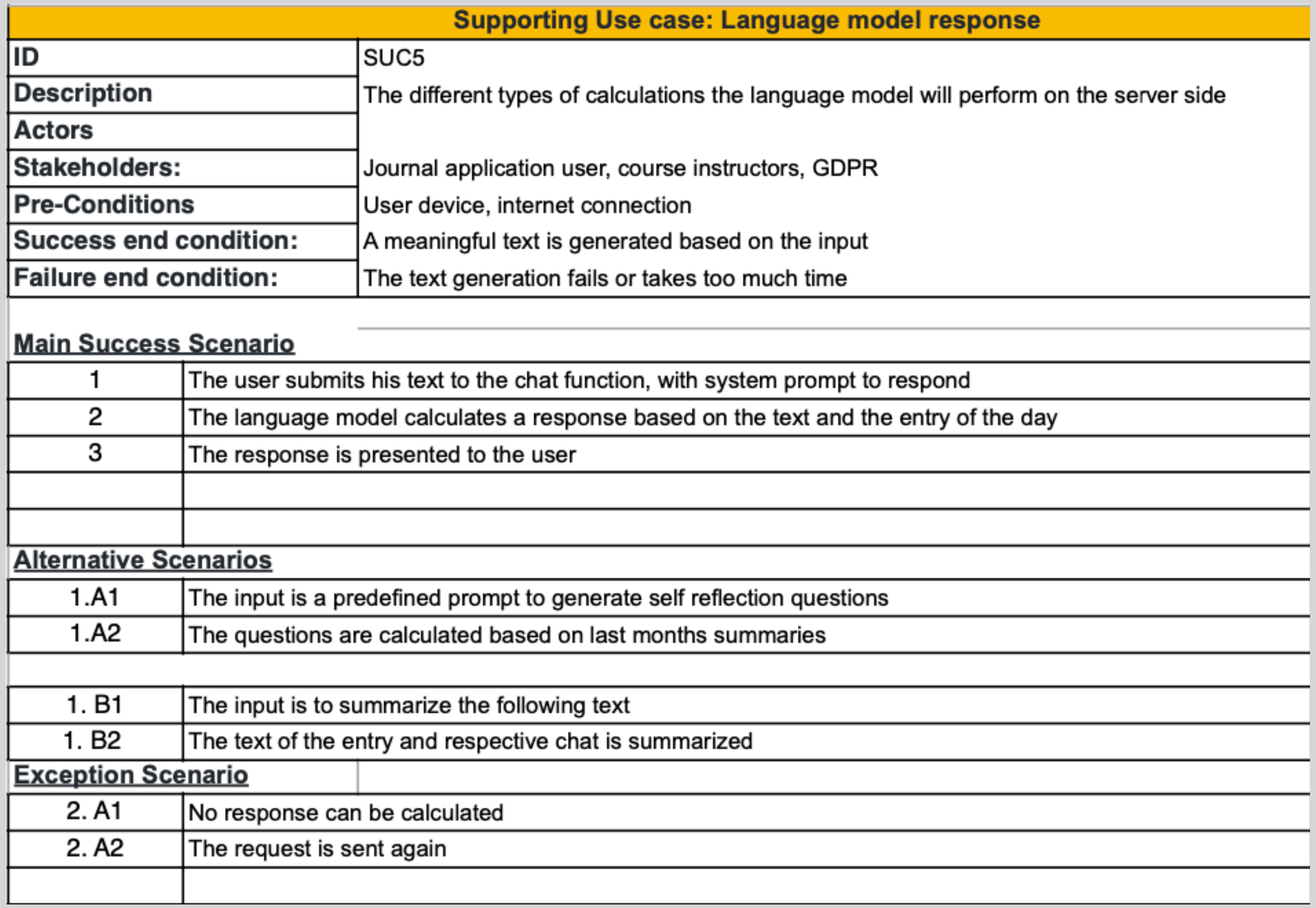
| Supporting Use case: Send data to server | |
|---|---|
| **ID** | SUC3 |
| **Description** | Send collected data from the user device to the server and stores in a database |
| **Actors** | |
| **Stakeholders:** | Journal application user, course instructors, GDPR |
| **Pre-Conditions** | User device, internet connection |
| **Success end condition:** | The data is correctly entered in the database after being transmitted to the server |
| **Failure end condition:** | The data cannot be entered in the database |
| **Main Success Scenario** | |
| 1 | The user finishes an entry, chat or reflection |
| 2 | The data is sent to the server and entered in the database |
| | |
| | |
| | |
| **Alternative Scenarios** | |
| **Exception Scenario** | |
| 2.A1 | The data cannot be sent due to internet connection |
| 2.A2 | The data is stored locally and synchronized at a later time |

✓ Implemented



| Supporting Use case: Summarize data | |
|---|---|
| **ID** | SUC4 |
| **Description** | Data in the database is summarized for long term storage and reference |
| **Actors** | |
| **Stakeholders:** | Journal application user, course instructors, GDPR |
| **Pre-Conditions** | |
| **Success end condition:** | The data is summarized and successfully stored in the database again |
| **Failure end condition:** | The data cannot be summarized |
| **Main Success Scenario** | |
| 1 | At nighttime the unsummarized data in the database is selected |
| 2 | The selected data is summarized using a language model |
| 3 | The summary is stored in the database again |
| | |
| | |
| **Alternative Scenarios** | |
| 1. A1 | The user selects the chat functionality on the same day |
| 1. A2 | The entry of this day is summarized immediately |
| **Exception Scenario** | |
| 2.A1 | The data cannot be summarized |
| 2.A2 | The unprocessed data is left for the next nighttime cycle to be summarized |
| | |

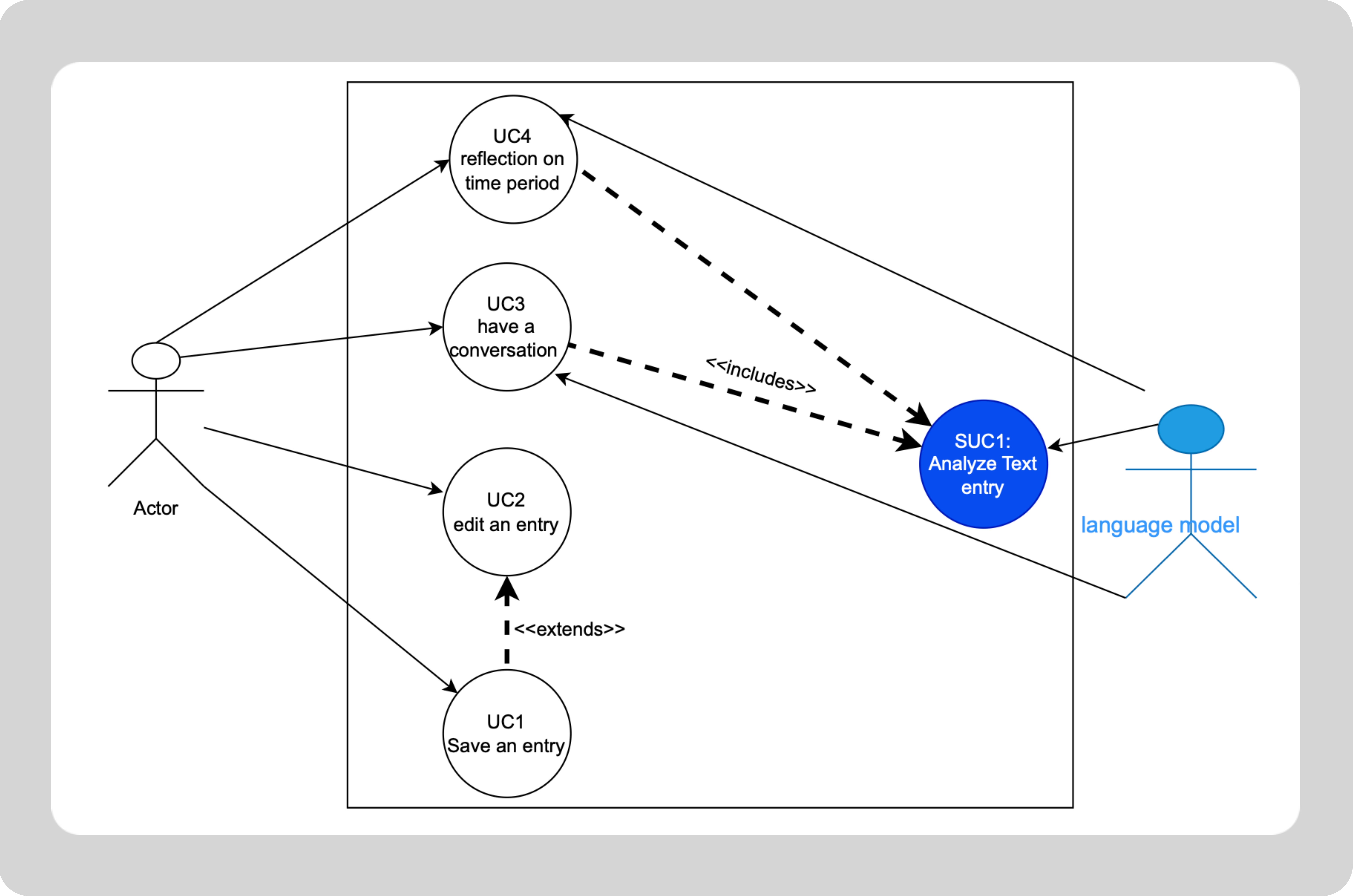
# Supporting use case 5



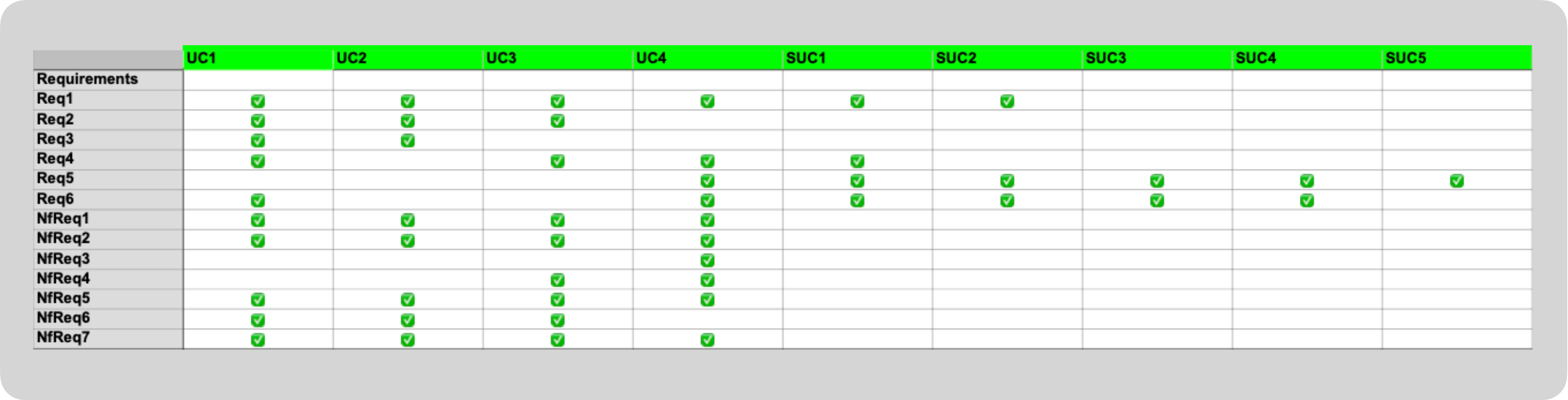
| Supporting Use case: Language model response | |
|---|---|
| **ID** | SUC5 |
| **Description** | The different types of calculations the language model will perform on the server side |
| **Actors** | |
| **Stakeholders:** | Journal application user, course instructors, GDPR |
| **Pre-Conditions** | User device, internet connection |
| **Success end condition:** | A meaningful text is generated based on the input |
| **Failure end condition:** | The text generation fails or takes too much time |
| | |
| **Main Success Scenario** | |
| 1 | The user submits his text to the chat function, with system prompt to respond |
| 2 | The language model calculates a response based on the text and the entry of the day |
| 3 | The response is presented to the user |
| | |
| | |
| **Alternative Scenarios** | |
| 1.A1 | The input is a predefined prompt to generate self reflection questions |
| 1.A2 | The questions are calculated based on last months summaries |
| | |
| 1. B1 | The input is to summarize the following text |
| 1. B2 | The text of the entry and respective chat is summarized |
| **Exception Scenario** | |
| 2. A1 | No response can be calculated |
| 2. A2 | The request is sent again |
| | |

✓ Implemented

# Use case diagram

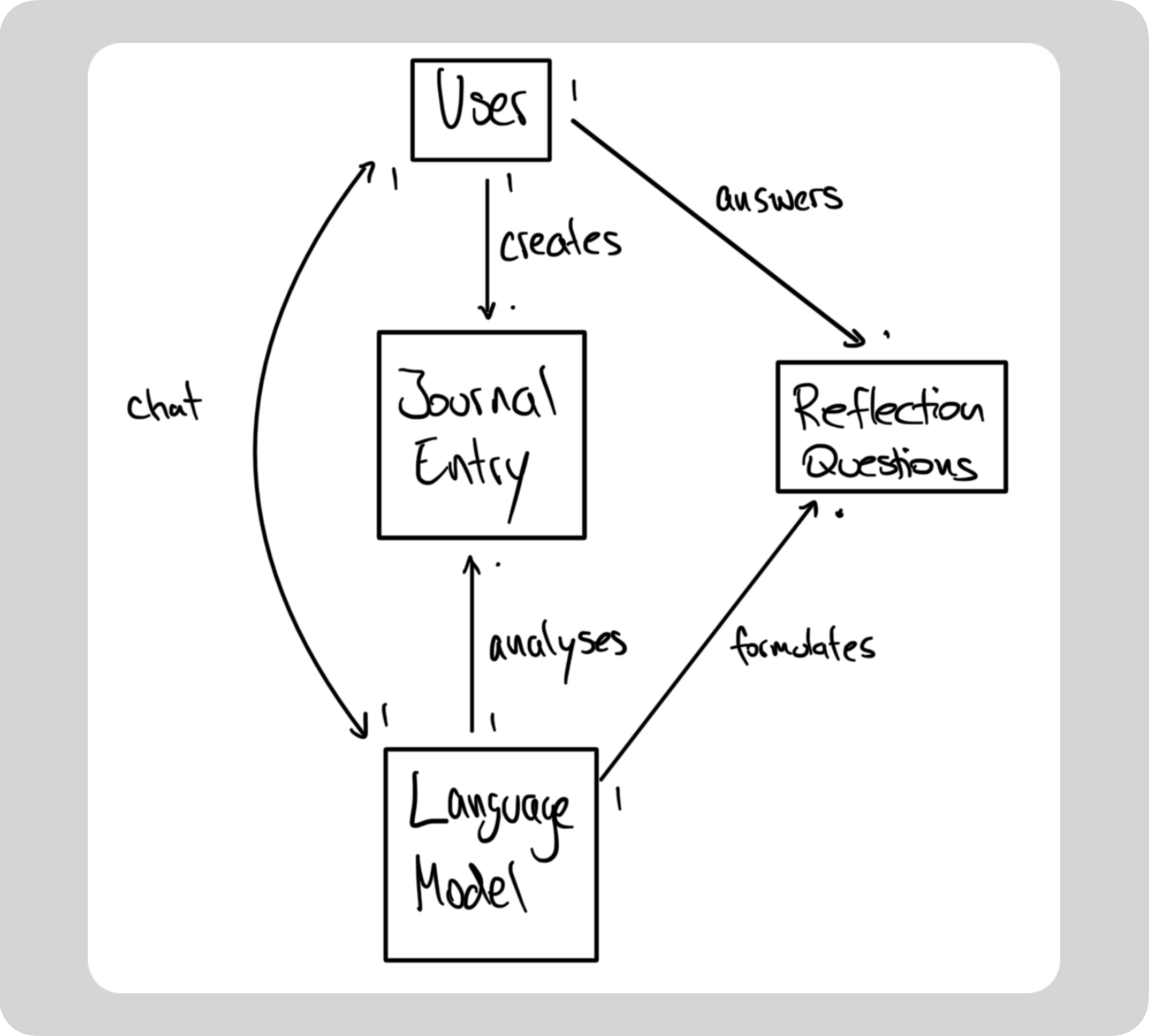


# Traceability matrix

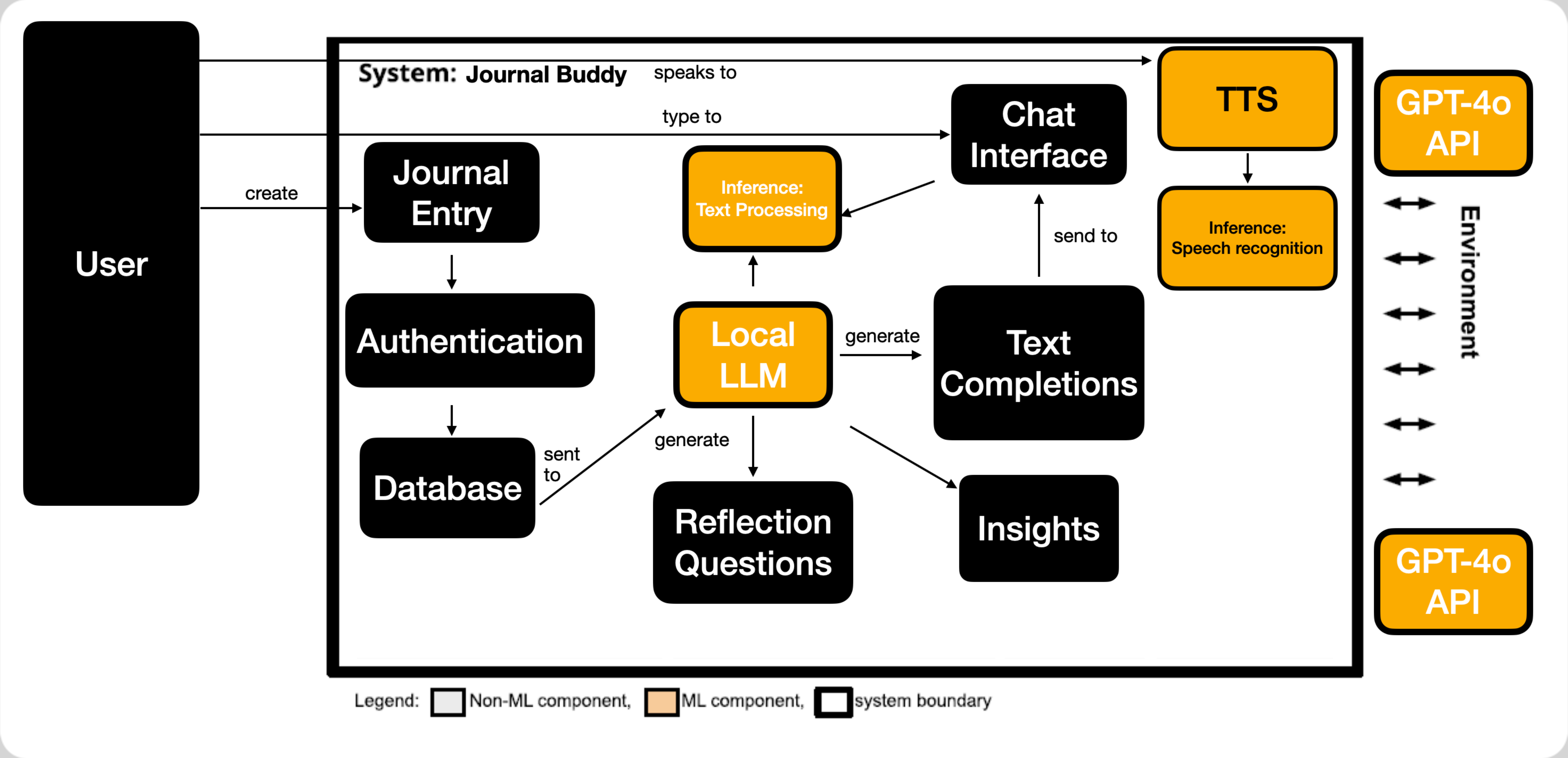


| | UC1 | UC2 | UC3 | UC4 | SUC1 | SUC2 | SUC3 | SUC4 | SUC5 |
|---|---|---|---|---|---|---|---|---|---|
| Requirements | | | | | | | | | |
| Req1 | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | | | |
| Req2 | ✅ | ✅ | ✅ | | | | | | |
| Req3 | ✅ | ✅ | | | | | | | |
| Req4 | ✅ | | ✅ | ✅ | ✅ | | | | |
| Req5 | | | | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| Req6 | ✅ | | | ✅ | ✅ | ✅ | ✅ | ✅ | |
| NfReq1 | ✅ | ✅ | ✅ | ✅ | | | | | |
| NfReq2 | ✅ | ✅ | ✅ | ✅ | | | | | |
| NfReq3 | | | | ✅ | | | | | |
| NfReq4 | | | ✅ | ✅ | | | | | |
| NfReq5 | ✅ | ✅ | ✅ | ✅ | | | | | |
| NfReq6 | ✅ | ✅ | ✅ | | | | | | |
| NfReq7 | ✅ | ✅ | ✅ | ✅ | | | | | |

# Domain model



# Architecture diagram



# Components description

**Authentication:** this component allows the user to get access to the services of the app. This also allows for the user to add, remove edit or delete entries while using the app. **Req<1>**

**Chat Interface:** this components is the GUI that facilitates the interaction between the user and other components of the system. **<Req>2**

**Database:** this component allows uses to store and retrieve entries. Entries stored in this database are used by both the local LLM and online GPTs for gaining better context about the user. **Req<3>**

**Local LLM:** a large language model that runs locally on the users device to aid inference where the user selects it or it is automatically selected when there is no or slow network to connect to the GPTs. It is also retrained for better inference on the user **<Req>4**

**Reflection:** this component takes the user entries for a specific time period and summarizes key point for the user. **Req5**

**Text completions:** this component is the main feature of the system. There system makes inference based on the user entries and returns text to keep up a conversation with the user. **<Req>6**

# Design questions and answers

**1. How do we ensure the privacy of journal entries?**
- Implement strong encryption for data storage and secure transmission protocols to ensure that journal entries remain confidential.

**2. What kind of natural language understanding capabilities are required to interpret journal entries effectively?**
- The system needs to handle a wide variety of language nuances, including sentiment analysis, intent recognition, and contextual understanding.

**3. How should the system handle misunderstood or ambiguous journal entries?**
- The system should ask clarifying questions to ensure accurate understanding and response, improving interaction quality.

**4. What criteria will be used to determine the effectiveness of the AI's journal entry analysis?**
- Effectiveness can be measured by the relevance and helpfulness of the AI’s responses, as well as the user’s engagement level and feedback.

**5. What are the response time goals for the AI, and how will these be achieved?**
- Aim for a maximum response time of 5 seconds, as outlined in the requirements, by optimizing the AI model and infrastructure for speed.

**6. How will the system handle multiple languages if future expansion requires it?**
- Plan for a modular language processing design (local and API) that allows easy integration of additional language models as needed.

**7. How will the user interface design facilitate easy journal entry and interaction?**
- Design a clean, intuitive interface that prioritizes ease of use, with features like voice-to-text transcription and easy navigation.

**8. How can the system personalize conversations based on past interactions without compromising privacy?**
- Utilize on-device LLM processing option to personalize interactions while keeping the data localized, ensuring privacy.

**9. How will the system handle user corrections or disagreements with the AI’s understanding or responses?**
- Allow users to provide feedback on specific responses, which can be used to refine the AI model and improve accuracy.

**10. How will JournalFriend integrate with other platforms or services, if required?**
- Plan for API development that allows seamless integration with other health and image platforms while maintaining privacy and security standards.