3rd meeting with the customer

Date: 28/02/2025

Key Discussion Points

1. **Use Case Diagram Review**
   - Have all main actors been considered?
   - No remarks; everything looks good.
2. **Integration with External Platforms**
   - Competitions will take place on another platform.
   - Should integration be automatic, or is a manual process sufficient?
   - Companies currently submit competition links via Google Forms.
   - Possible solution: Implement a parser for Google Forms.
   - Can a company manually input competition data instead?
   - Research possible integration methods for Google Forms.
3. **Competition Management**
   - Can companies delete competitions?
     - Yes, but with an additional verification step.
   - Can competitions be edited or deleted after they have started?
     - No modifications allowed once the competition begins.
   - Can students edit their submissions?
     - Students can submit only once before the deadline.
   - Competition status tracking:
     - Should this be automated or manually updated by the company?
     - Statuses should be strictly time-based, as set by the client.
     - Deadline extensions are not considered.
4. **Backend & Database Considerations**
   - Can two platforms share a common database and API?
     - No objections; this is feasible.
   - Should data replication be implemented?
     - Possible if it is not too time-consuming.
5. **Jury Role & Functionality**
   - Should jury members have access to the platform where evaluations are done?
     - They can log in via their personal accounts and access tasks in the evaluation system (Brics).

- Can a shared authentication system be implemented?
  - Yes, this is possible.
- Who grants access to competition submissions for the jury?
  - Organizations manage jury access.
- Can jury members participate in competitions they are judging?
  - No, judges cannot compete in the events they evaluate.
- Should we anonymize student data for jury members?
  - Yes, anonymization should be implemented, but companies should retain access to full data.
- Are scores automatically applied, or should companies review them first?
  - Companies should approve jury scores before they are finalized.
- Should jury members sign NDAs or other agreements?
  - This is the responsibility of the companies.
- If a jury member fails to evaluate assigned competitions, who steps in?
  - The company should take over evaluation duties.
- How does a user become a jury member?
  - Must have participated in at least 15 competitions and reached level 30.
- Should there be a selection process for jury members?
  - Organizations can choose from a list of eligible jury candidates.

6. **Communication & Miscellaneous**
   - Should we store Telegram contacts along with email and phone numbers?
     - Yes, but communication happens outside the platform.
   - Should penalties be introduced (e.g., point deductions)?
     - No penalties; students can only earn points, not lose them.
   - Is jury functionality part of the MVP?
     - Yes, but it is a lower priority and should be implemented after the core system.

Active Notes (Next Meeting Topics)

- Finalize integration strategy: Should we develop a Google Form parser or allow manual competition entries?
- Determine whether competition status updates should be automated or company-managed.
- Review data replication feasibility for backend.
- Clarify jury role: Should jury scoring always require company approval?
- Discuss anonymization implementation for jury evaluations.
- Evaluate whether the jury selection process needs additional filtering criteria.

code for parsing google form

```python
from googleapiclient.discovery import build
from google.oauth2 import service_account
from datetime import datetime

SERVICE_ACCOUNT_FILE = 'still-habitat-433108-e9-88abb8ef4b6e.json'
SPREADSHEET_ID = '167X_tpTLxFGiivXOWPj9gmOP7Af2gJuyZKlWP5bQGa0'
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']

creds =
service_account.Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE,
scopes=SCOPES)


def update_google_sheet_with_survey_data(unti_id, user_data):
    """
    Обновляет данные студента в Google Таблице по UNTI ID, записывая
технические навыки и мотивационные вопросы.
    """
    sheets_service = build('sheets', 'v4', credentials=creds)
    session = Session1()
    student = session.query(Student).filter(Student.UNTId == unti_id).first()
    if student:
        update_survey_status(student.TgId, "tech", "pending_review")
        update_survey_status(student.TgId, "motivation", "pending_review")

    try:
        # Проверяем, есть ли UNTI ID
        if not unti_id:
            print("❌ Ошибка: UNTI ID отсутствует!")
            return "❌ Ошибка: UNTI ID отсутствует!"

        # Загружаем строки таблицы
        result = sheets_service.spreadsheets().values().get(
            spreadsheetId=SPREADSHEET_ID,
            range="Лист1!A:AE"
        ).execute()
        rows = result.get('values', [])

        if not rows:
            print("❌ Таблица пустая или данные не загружены!")
            return

        header_row = rows[0]

        # Проверяем, есть ли столбец UNTI ID
        if "UNTI ID\n\nФормат: U123145" not in header_row:
```

```
            print("❌ Ошибка: Столбец UNTI ID не найден в таблице!")
            return

        unti_id_col_index = header_row.index("UNTI ID\n\nФормат: U123145")
        target_row_index = None

        # Поиск строки с UNTI ID
        for row_index, row in enumerate(rows[1:], start=2):  # Индексация в
Google Sheets с 1
            if len(row) > unti_id_col_index and row[unti_id_col_index] ==
unti_id:
                target_row_index = row_index
                break

        if target_row_index is None:
            print(f"❌ Студент с UNTI ID {unti_id} не найден в таблице.
Проверь регистрацию!")
            return "❌ Ошибка: Студент не найден."

        # ✅ **Создаём структуры данных, если их нет**
        if "tech_answers" not in user_data:
            user_data["tech_answers"] = {}

        if "motivation_answers" not in user_data:
            user_data["motivation_answers"] = {}

        # 📌 **Подготавливаем данные**
        update_data = {
            "N": "; ".join(user_data['tech_answers'].get("3D-Моделирование",
[])),
            "O": "; ".join(user_data['tech_answers'].get("Радиоэлектроника",
[])),
            "P": "; ".join(user_data['tech_answers'].get("Программирование",
[])),
            "Q": "; ".join(user_data['tech_answers'].get("Работа с
композитами", [])),
            "R": "; ".join(user_data['tech_answers'].get("Полёты на дронах",
[])),
            "S": "; ".join(user_data['tech_answers'].get("Производство/сборка
БПЛА", [])),
            "T": user_data["motivation_answers"].get("Чему хотели бы научиться
в этом сезоне?", "-"),
            "U": user_data["motivation_answers"].get(
                "Как узнали о программе? (реклама, от знакомого, на паре
рассказали и т.д.)", "-"),
            "V": user_data["motivation_answers"].get("Если вас кто-то
```

```
пригласил, укажите его ФИО и TG-никнейм.", "-"),
            "W": user_data["motivation_answers"].get("Какая цель? (Кратко: не
более 1 предложения)", "-"),
            "X": user_data["motivation_answers"].get("Почему решили
участвовать?", "-"),
            "Y": user_data["motivation_answers"].get("Каким специалистом
хотите быть?", "-"),
            "Z": user_data["motivation_answers"].get(
                "Кратко расскажите о себе (студент, работник - где?).
Расскажите о ваших увлечениях и хобби.", "-"),
        }

        update_range = f"Лист1!N{target_row_index}:Z{target_row_index}"
        body = {'values': [[update_data[col] for col in
sorted(update_data.keys())]]}
        # Запись данных в таблицу
        sheets_service.spreadsheets().values().update(
            spreadsheetId=SPREADSHEET_ID,
            range=update_range,
            valueInputOption="RAW",
            body=body
        ).execute()
        print(f"✅ Данные успешно обновлены в строке {target_row_index} (UNTI
ID: {unti_id})!")
        return "✅ Данные успешно сохранены в Google Таблицу."

    except Exception as e:
        print(f"❌ Ошибка при обновлении Google Таблицы: {e}")
        return f"❌ Ошибка: {e}"
```

use case diagram