ДОДАТОК А

Графічні матеріали

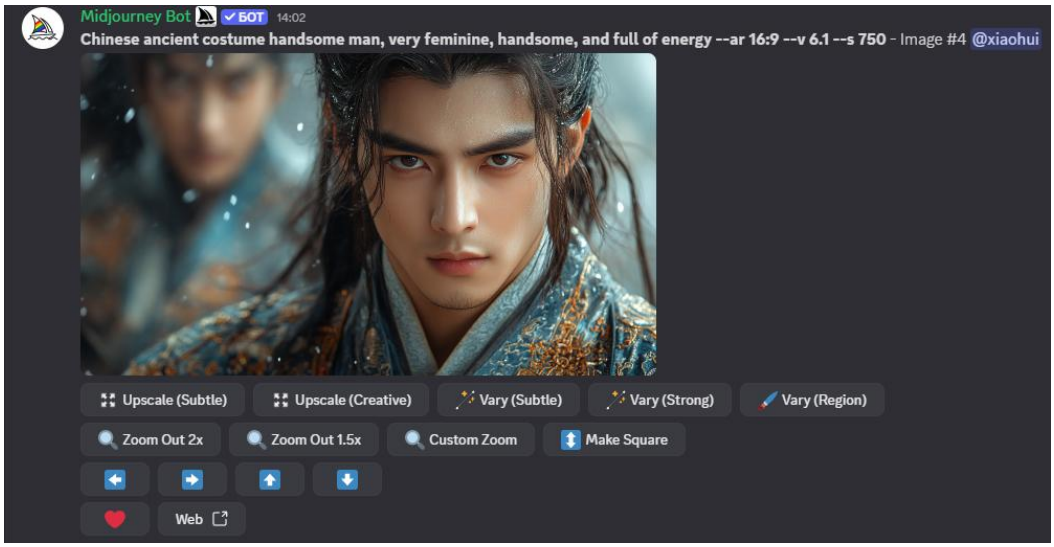

Рисунок А.1 – Запит і результат виконання запиту в системі генерації зображень Midjourney

Рисунок А.2 – Інтерфейс Adobe Photoshop для генеративного редагування зображень

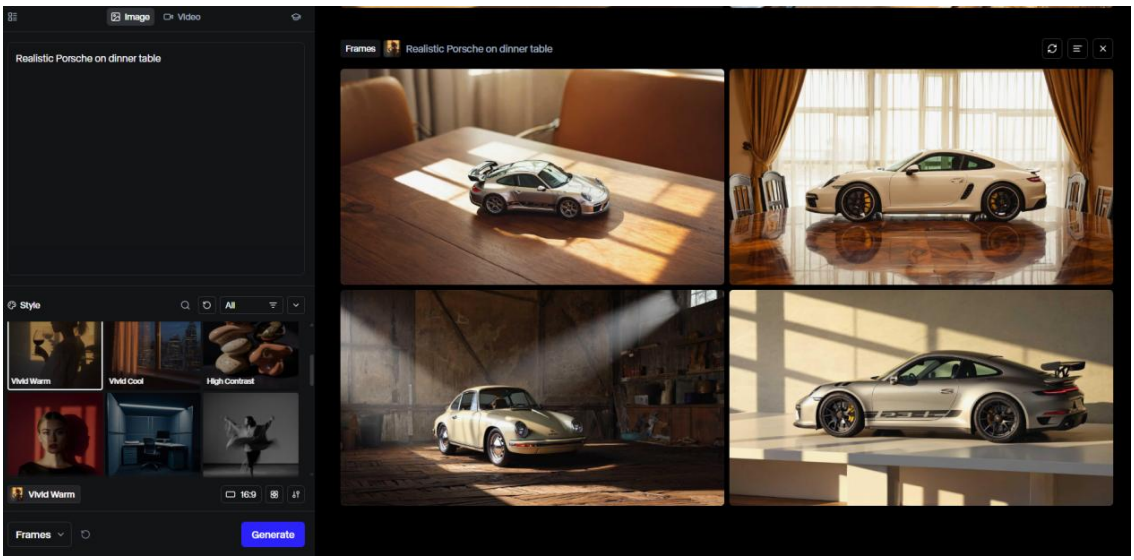

Рисунок А.3 – Інтерфейс Runway AI для генерації зображень

Рисунок А.4 – Інтерфейс Runway AI редагування зображення

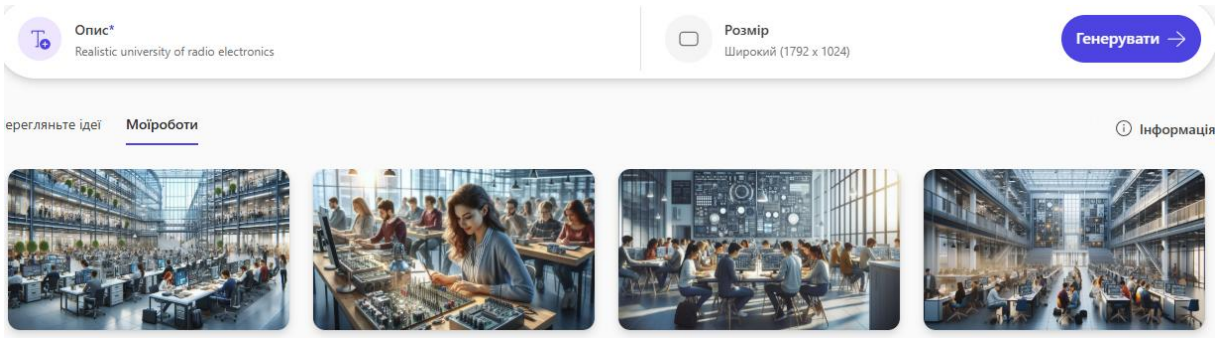

Рисунок А.5 – Інтерфейс Microsoft Designer для генерації зображень

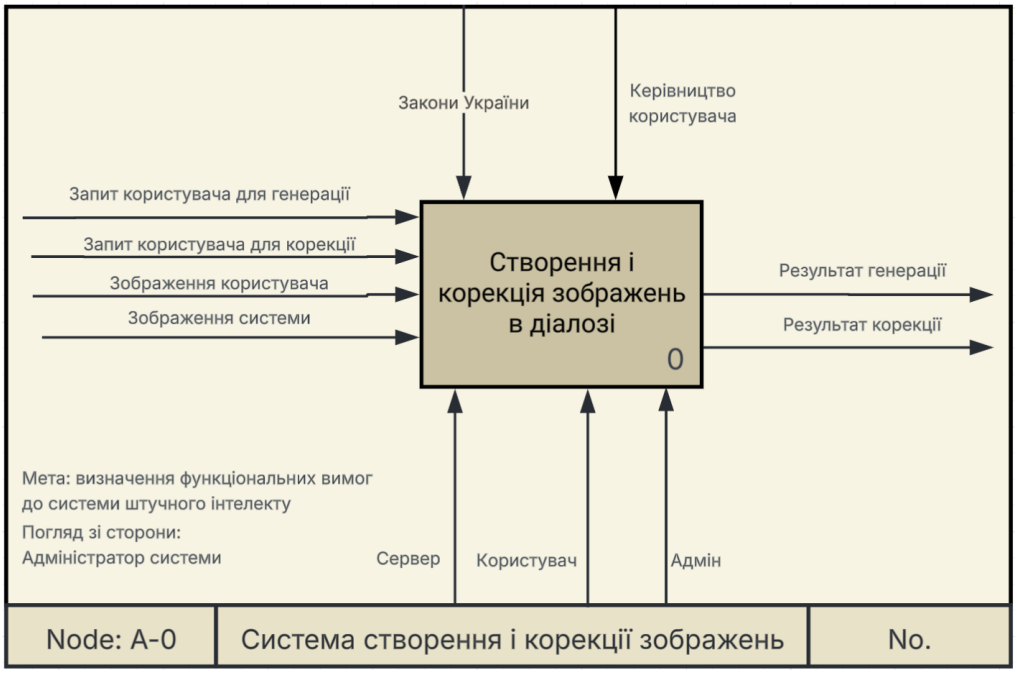

Рисунок А.6 – Концептуальна діаграма для системи створення і корекції зображень в діалозі

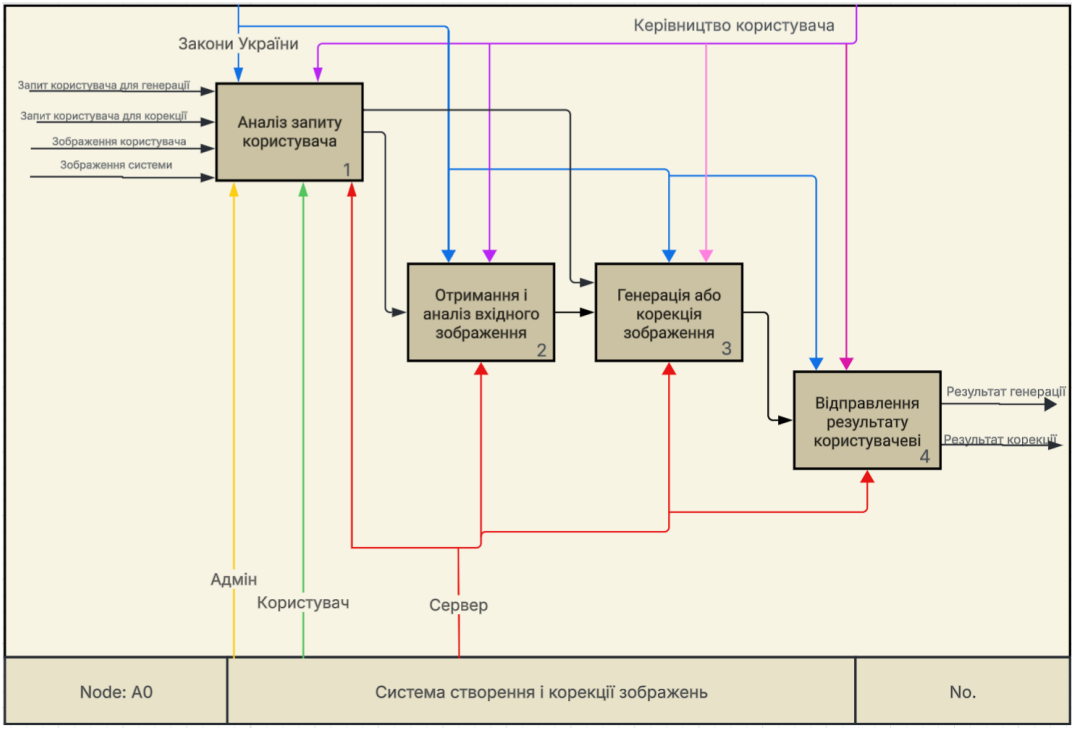

Рисунок А.7 – Перший рівень декомпозиції для системи створення і корекції зображень в діалозі

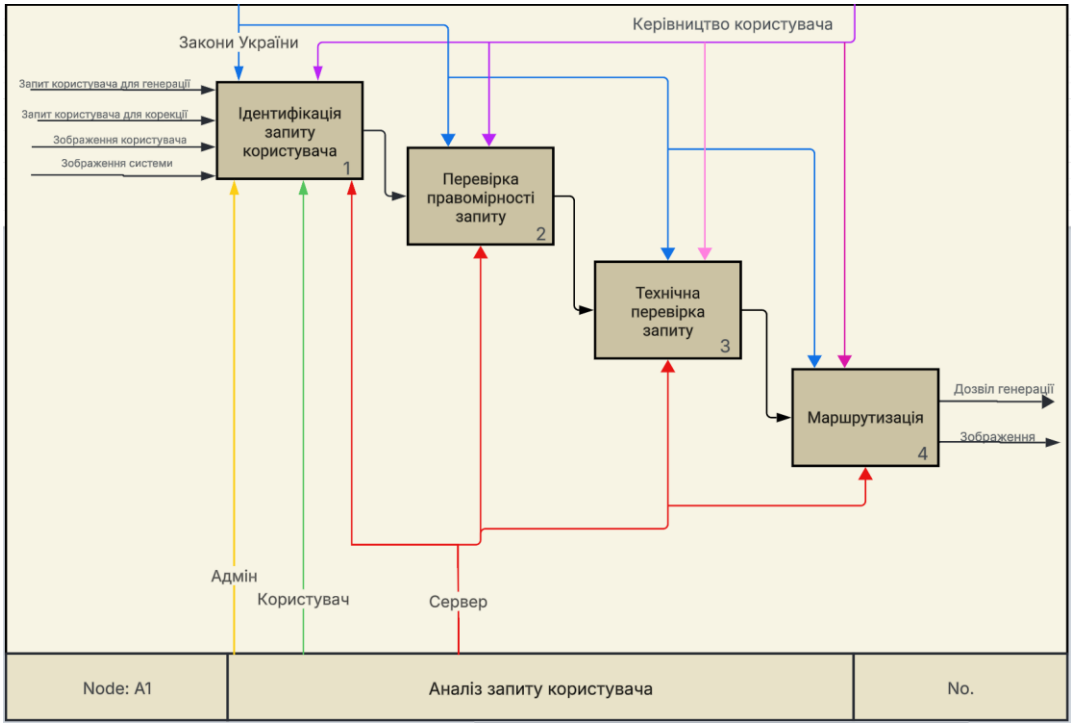

Рисунок А.8 – Декомпозиція бізнес–процесу «Аналіз запиту користувача»

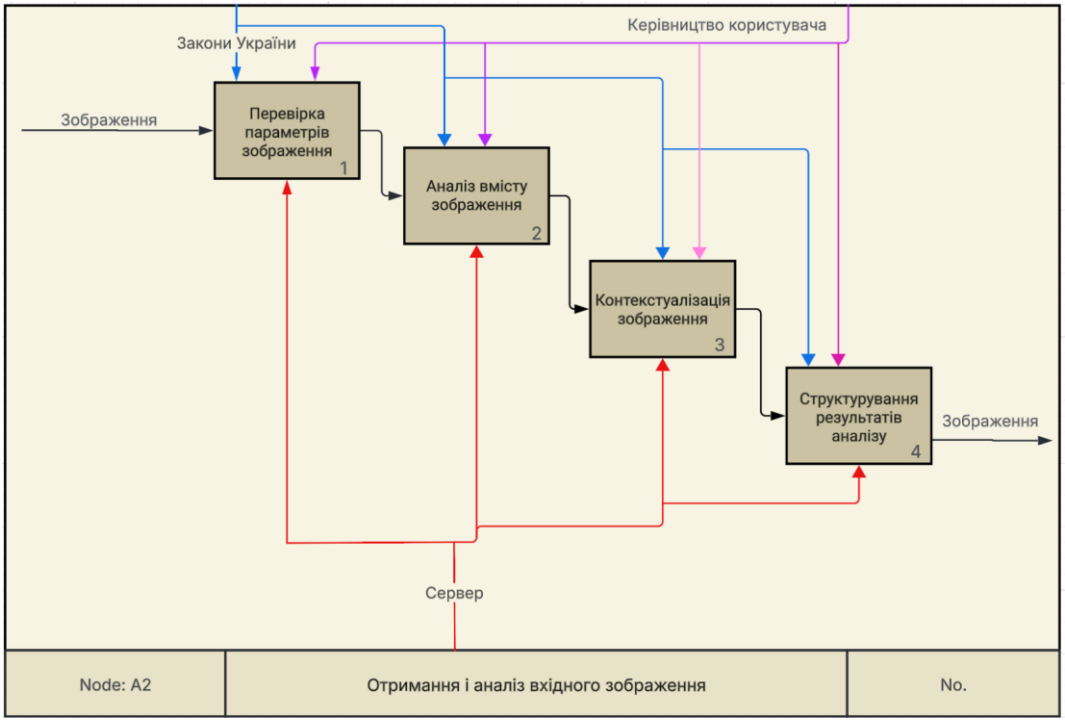

Рисунок А.9 – Декомпозиція бізнес–процесу «Отримання і аналіз вхідного зображення»

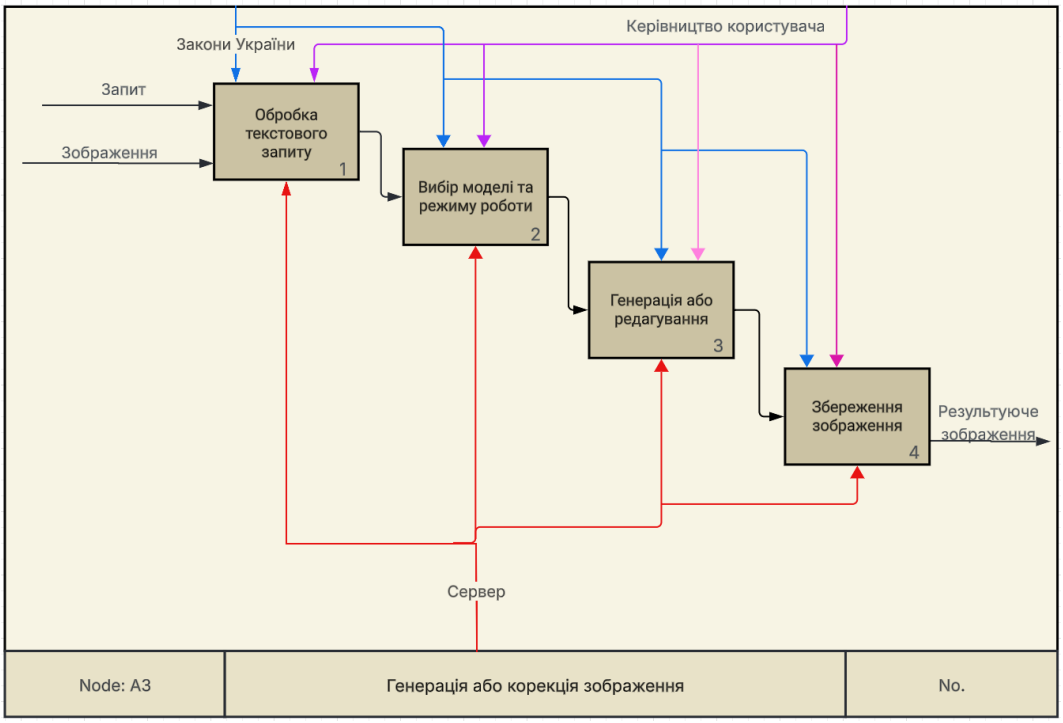

Рисунок А.10 – Декомпозиція бізнес–процесу «Генерація або корекція зображення»

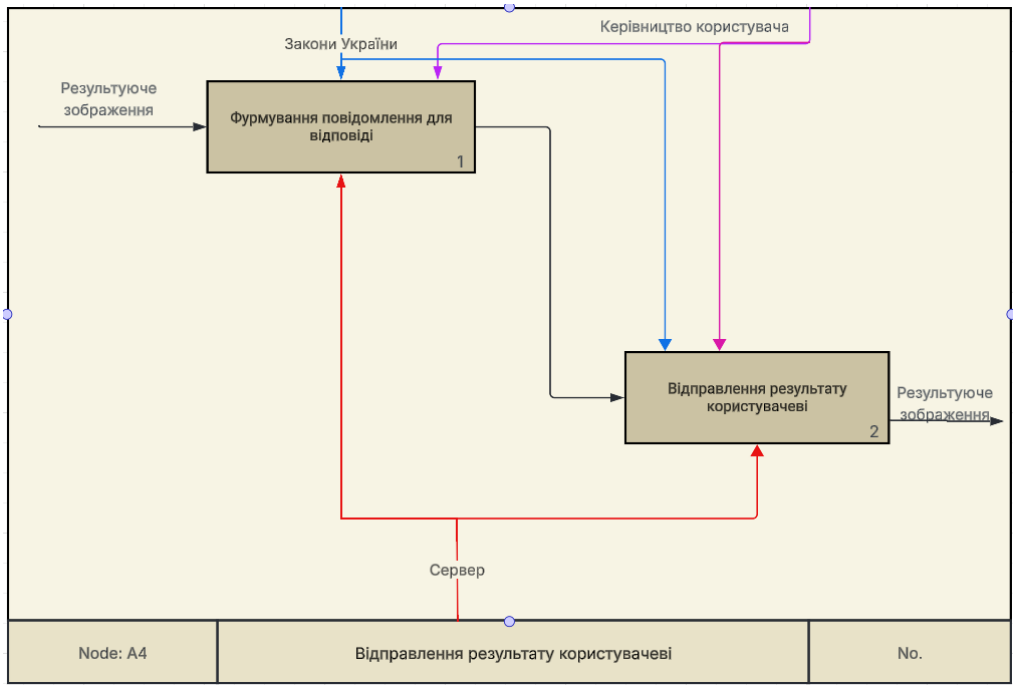

Рисунок А.11 – Декомпозиція бізнес–процесу «Відправлення результату користувачеві»

Рисунок А.12 – Діаграма потоків даних для системи створення і корекції зображень в діалозі

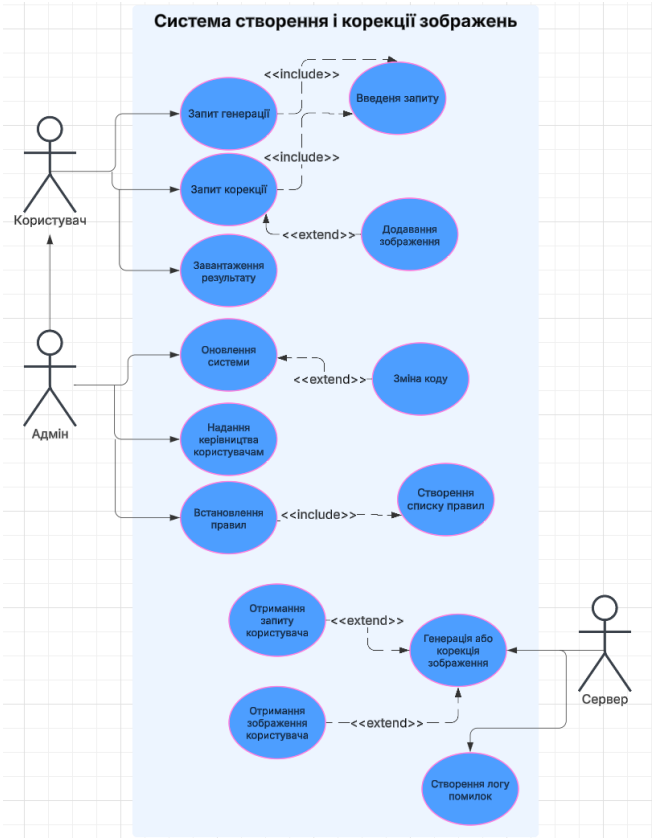

Рисунок А.13 – Діаграма варіантів використання для системи штучного інтелекту для створення і корекції зображень у діалозі

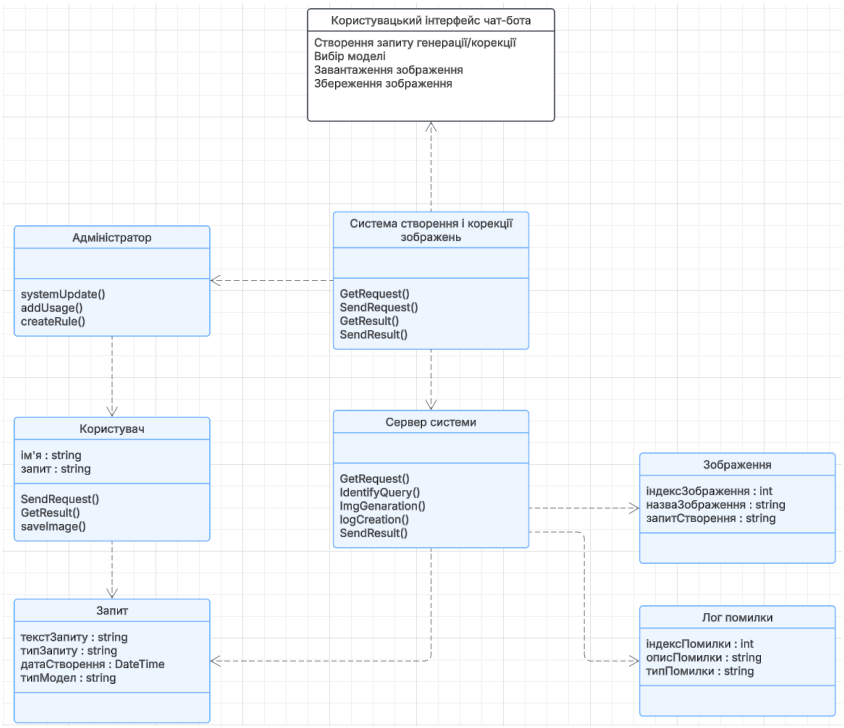

Рисунок А.14 – Діаграма класів для системи створення і корекції зображень за допомогою текстових підказок в діалозі

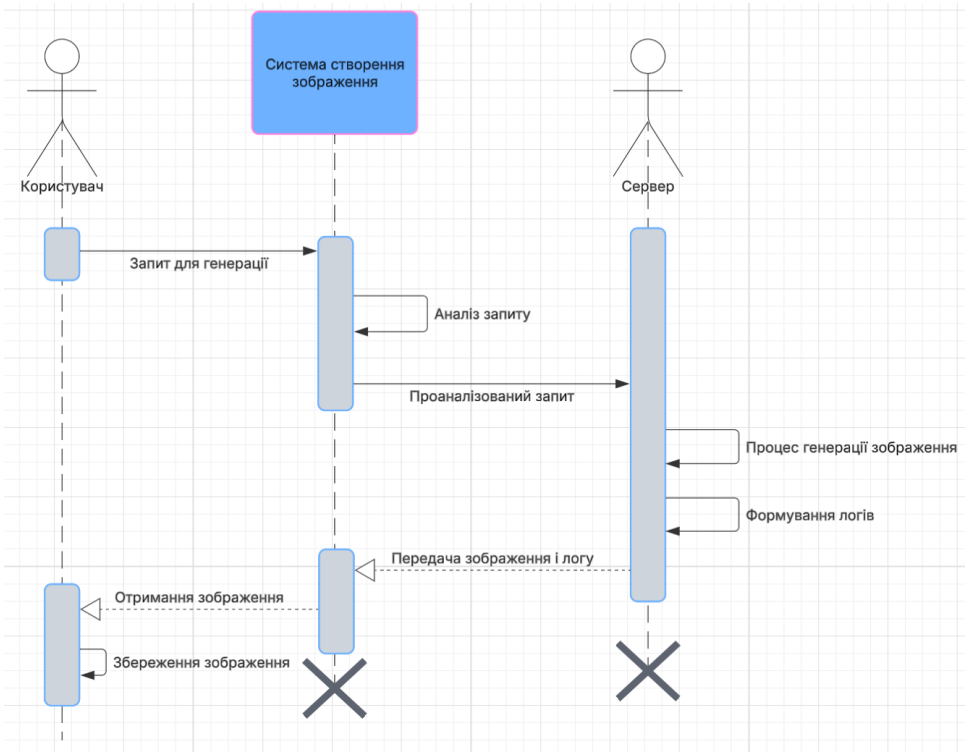

Рисунок А.15 – Діаграма послідовності дій для прецеденту «Створення зображення»

Рисунок А.16 – Карта алгоритму роботи системи створення і корекції зображень в діалозі

Рисунок А.17 – Початок роботи бота

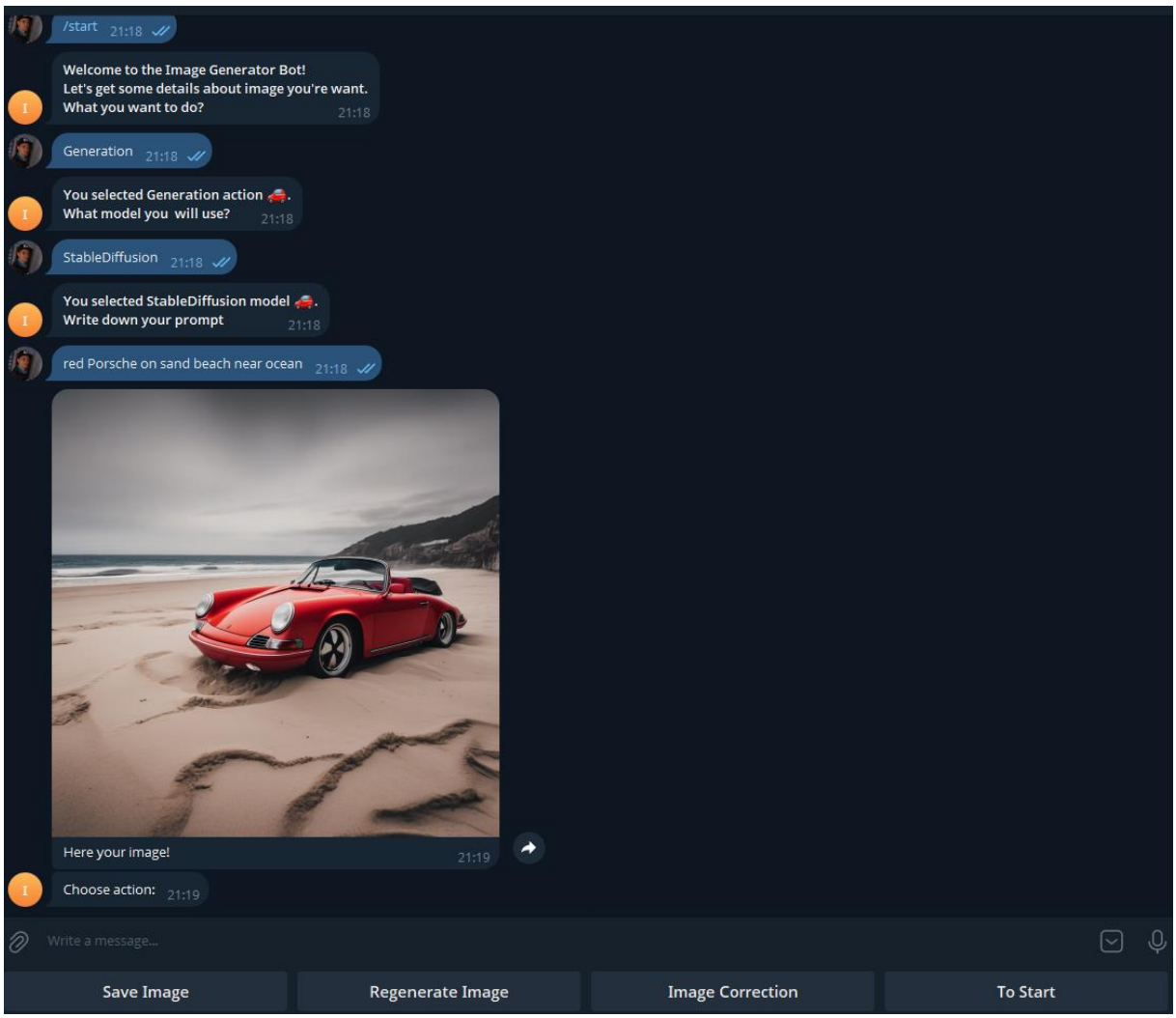

Рисунок А.18 – Тестування створення зображення

Рисунок А.19 – Тестування збереження створеного зображення

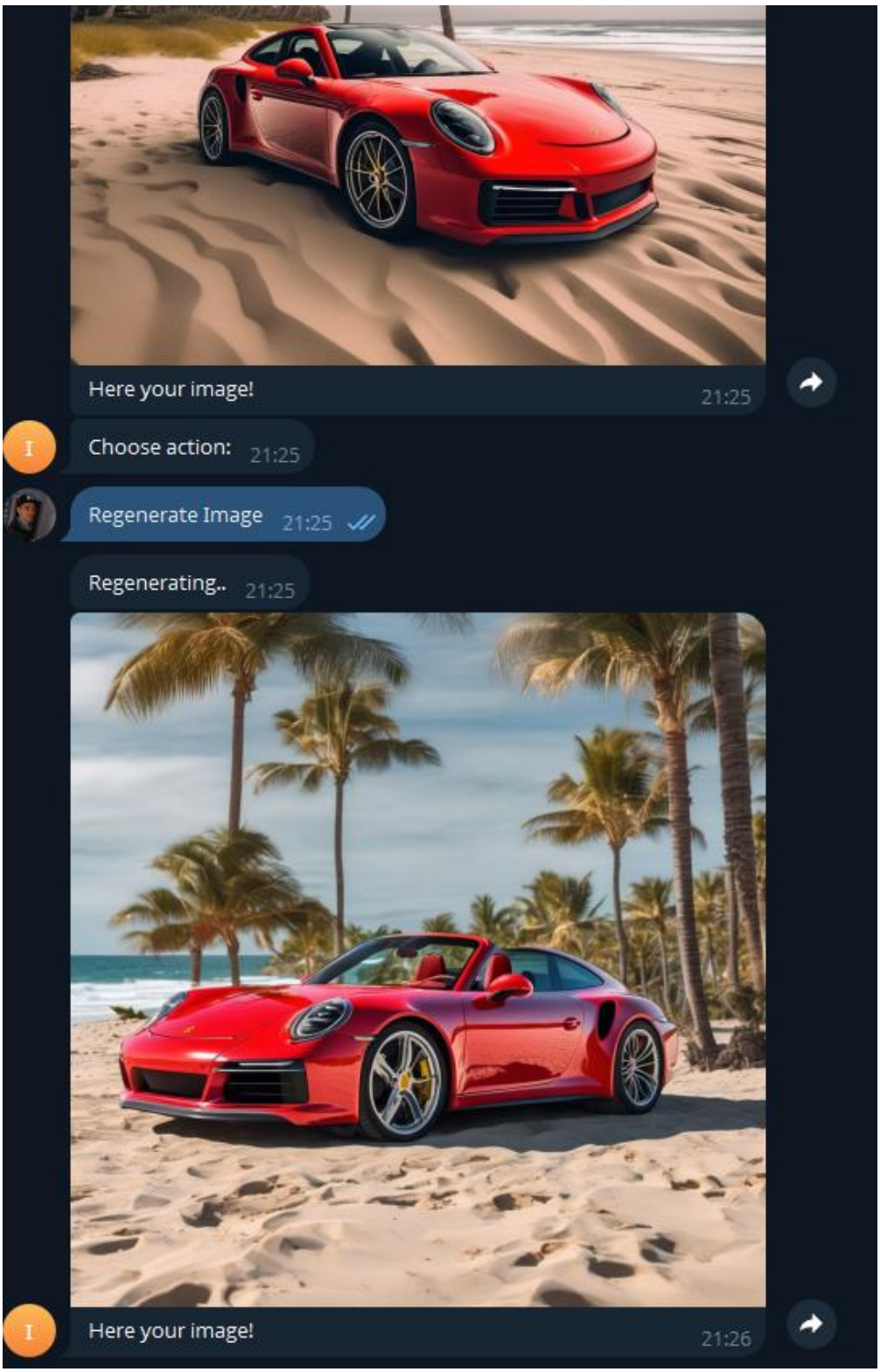

Рисунок А.20 – Тестування повторного створення зображення

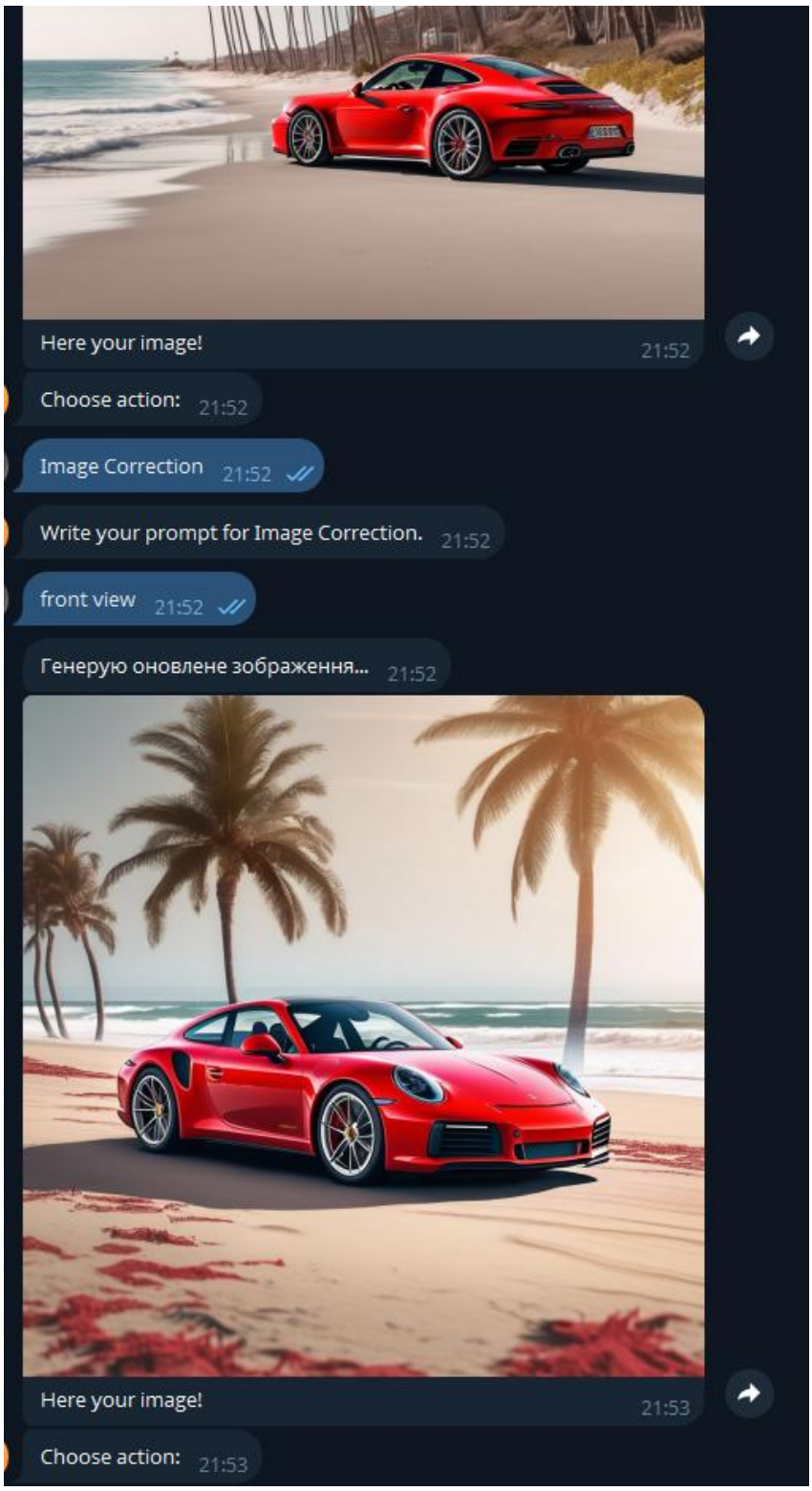

Рисунок А.21 – Тестування корекції для створеного зображення

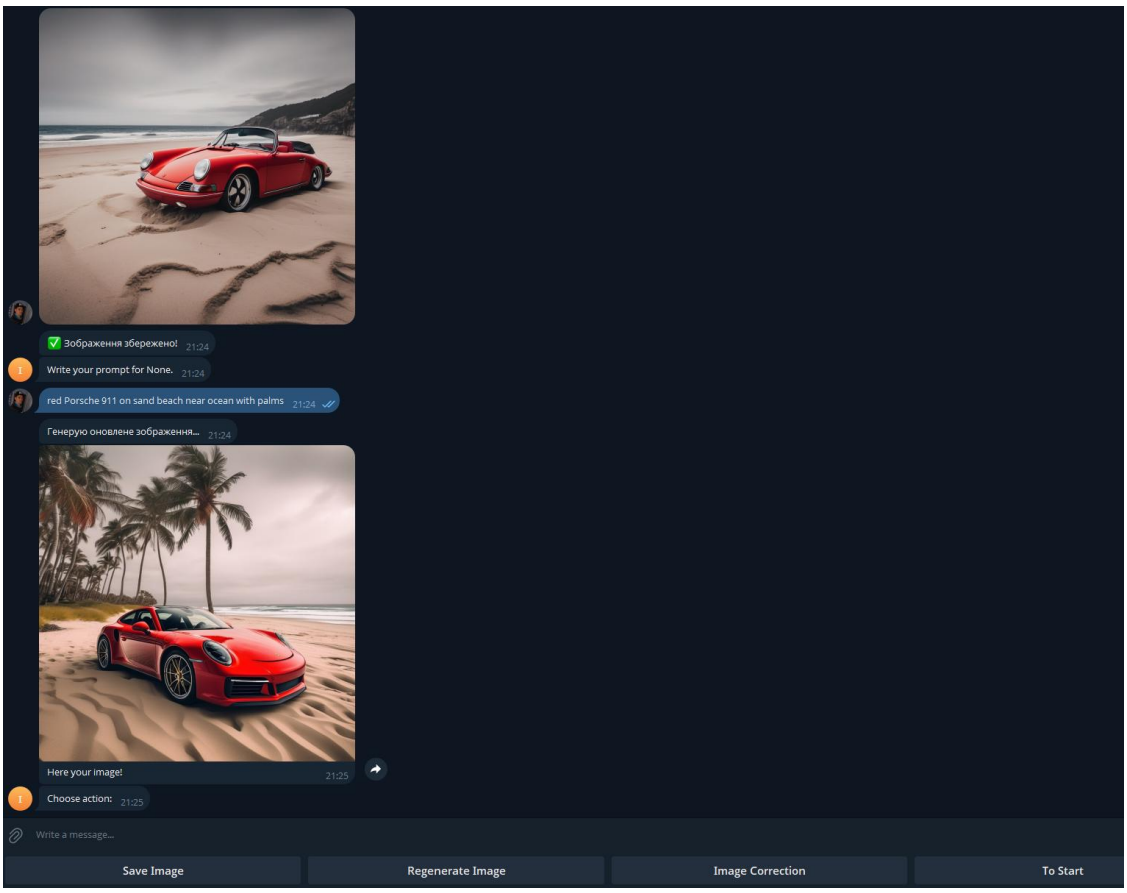

Рисунок А.22 – Тестування корекції зображення

ДОДАТОК Б

Керівництво користувача

Робота системи штучного інтелекту для створення і корегування починається, коли користувач заходить в чат і натискає кнопку «Start» і тоді бот відправляє користувачеві вітальне повідомлення і пропонує обрати, яку послугу користувач хоче отримати: створення зображення чи корекцію. Початок роботи системи зображено на рисунку Б.1.

Рисунок Б.1 – Початок роботи бота

Коли користувач обирає створення зображення, то система пропонує обрати модель для генерації. На рисунку Б.2 зображено вибір моделі.

Рисунок Б.2 – Вибір моделі для генерації зображення

Коли користувач обрав модель, то система запрошує користувача, щоб той написав свій запит для створення зображення. На рисунку Б.3 зображено інтерфейс запиту.

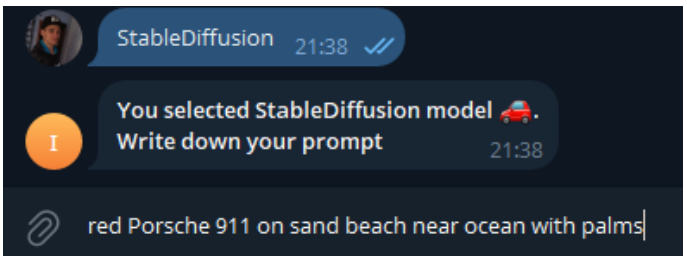

Рисунок Б.3 – Створення запиту користувача

Коли користувач натискає кнопку вводу, то система приймає запит користувача і на основі запиту починає генерувати зображення, після чого надсилає результат створення в чат. Додатково система пропонує користувачеві 4 функції: збереження, повторна генерація, корекція отриманого зображення і повернення на початок. Результат створення зображення наведено на рисунку Б.4.

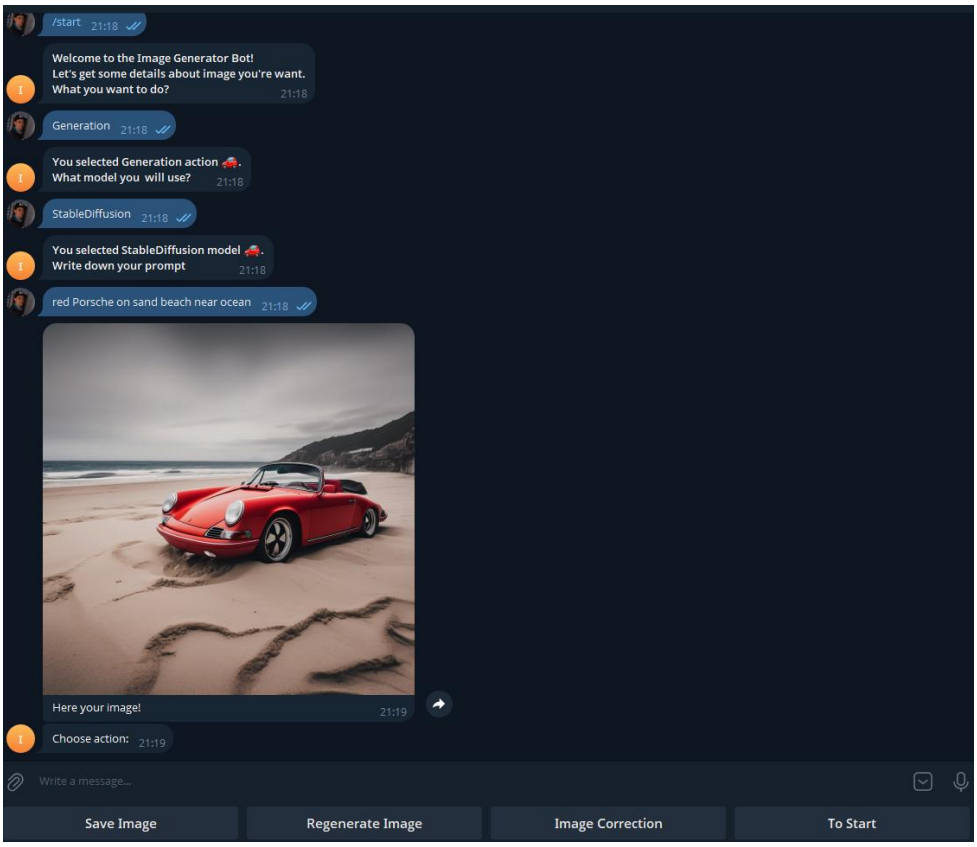

Рисунок Б.4 –Створення нового зображення

Коли користувач обирає збереження зображення, то система надсилає файл без стискання і втрат якості. Паралельно система автоматично повертається в стартове положення і пропонує власні функції. На рисунку Б.5 зображено результат збереження зображення.

Рисунок Б.5 –Збереження створеного зображення

Для повторного створення зображення користувач має пройти шлях генерації зображення і вийти на екран отримання результату генерації. Користувач натискає кнопку повторної генерації, і система, на основі збереженого запиту користувача, повторно генерує зображення. На рисунку Б.6 зображено повторне створення зображення.

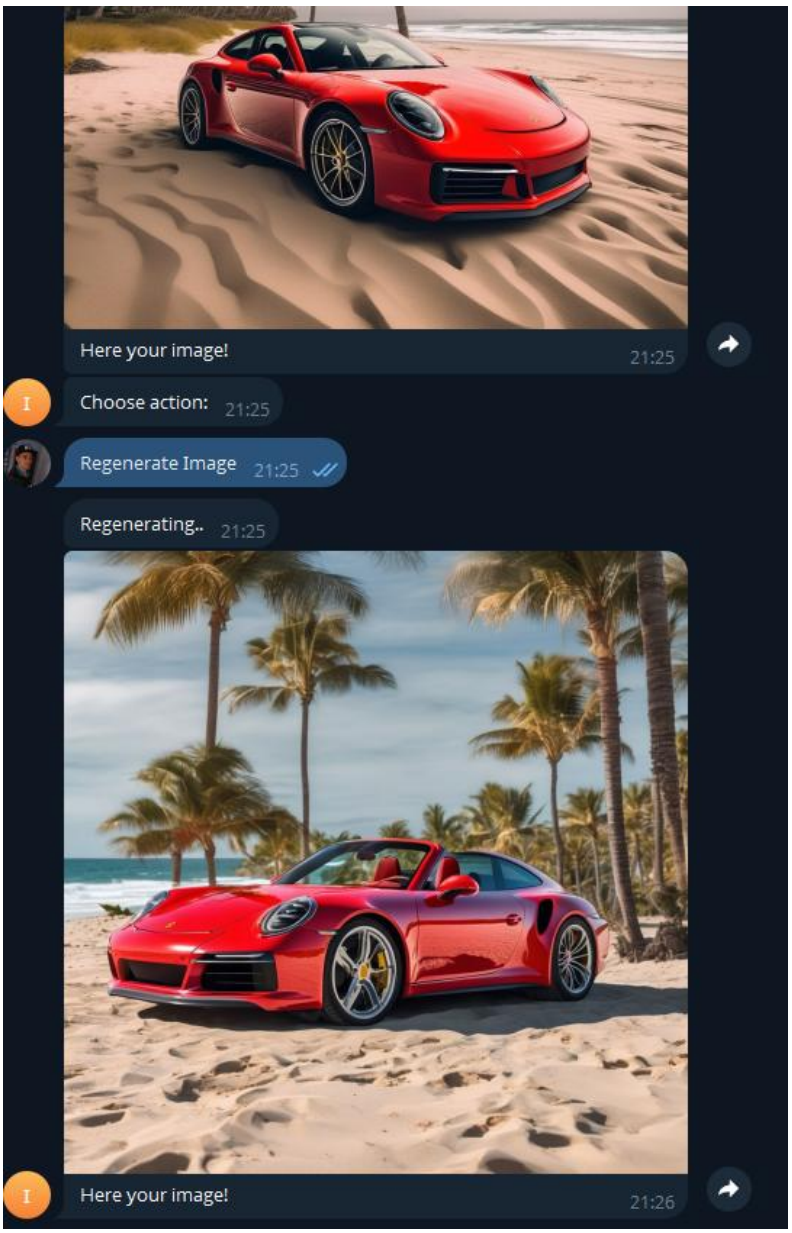

Рисунок Б.6 – Повторне створення зображення

На екрані отримання результату користувач може обрати функцію корекції зображення і система запросить додаткову інструкцію, після чого розпочнеться виконання корекції. Процес корекції для щойно створеного зображення наведено на рисунку Б.7.

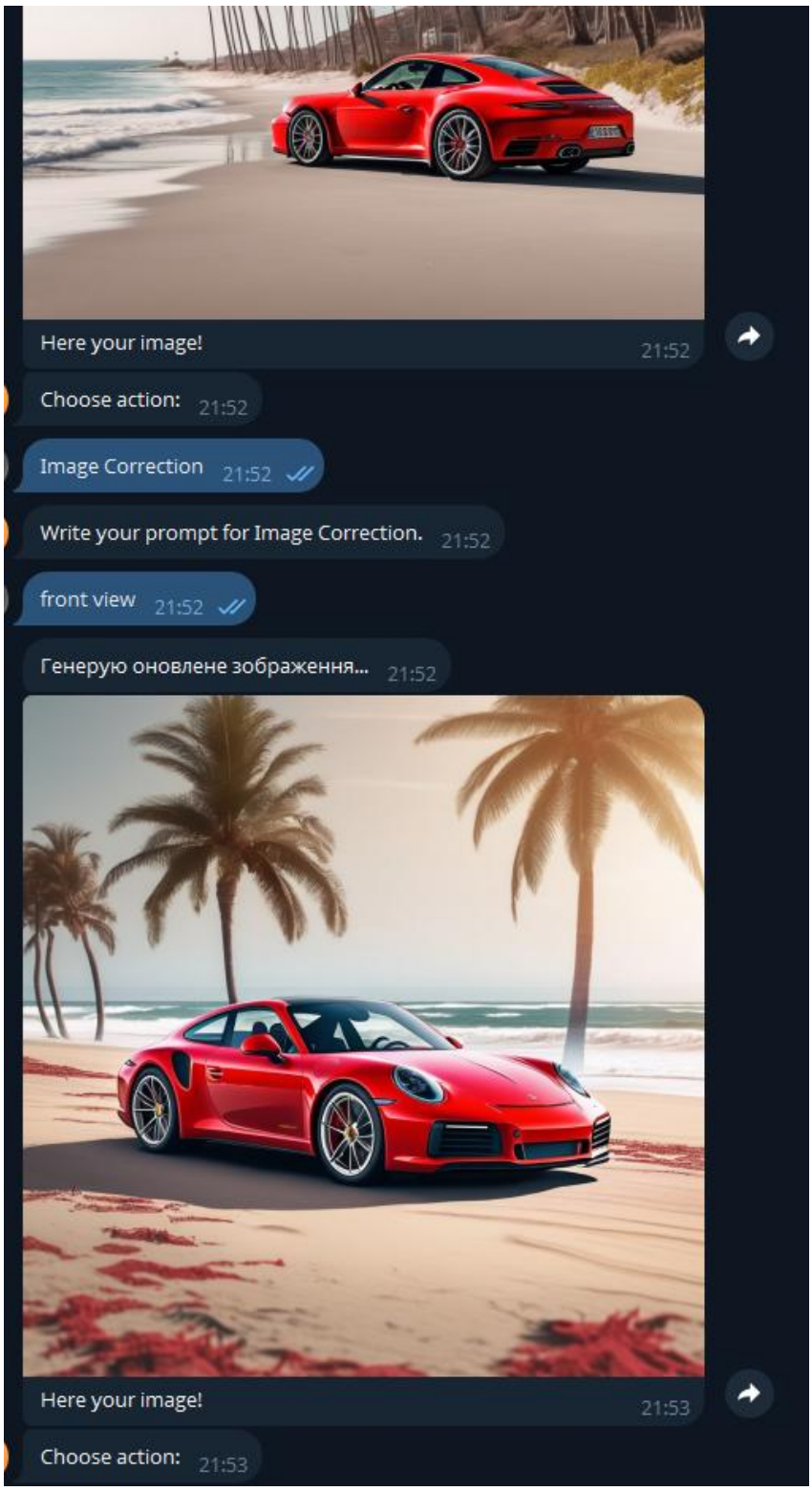

Рисунок Б.7 – Корегування створеного зображення

Для корекції власного зображення користувачеві необхідно повернутися в стартове положення програми і обрати корекцію, після чого система запросить користувача завантажити власне зображення. Це можна

зробити звичайною вставкою зображення, або за допомогою функції «Прикріпити файл». Після чого система попросить ввести запит, в якому користувач має описати контекст зображення і описати зміни, які він хоче отримати. Коли система скорегує і відправить зображення в чат, то користувач може знову згенерувати його повторно, або додати умови для корекції. На рисунку Б.8 зображено інтерфейс користувача після корекції власного зображення.

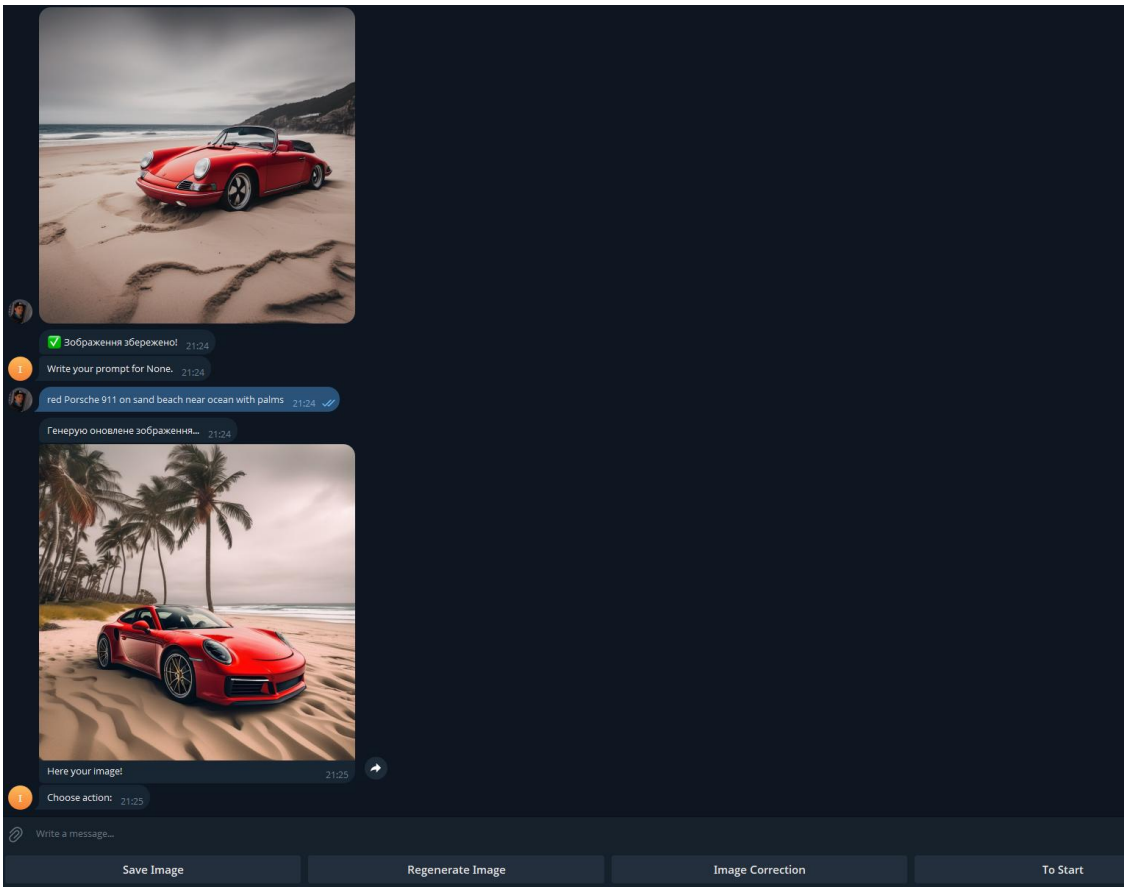

Рисунок Б.8 – Корегування власного зображення

ДОДАТОК В

Текст програми

Лістинг В.1 – Реалізація бізнес–процесу «Створення зображення»:

```
    # Функція генерації зображення
    def generate_image(prompt: str):
        # Токенізація тексту
        text_input = tokenizer(prompt, padding="max_length",
max_length=77, return_tensors="pt")
        input_ids = text_input.input_ids.to(device)
        # Отримання текстових ембеддингів
        text_embeddings = text_encoder(input_ids)[0]
        # Додаємо "порожній" запит для CFG
        uncond_input = tokenizer([""], padding="max_length",
max_length=77, return_tensors="pt")
        uncond_embeddings =
text_encoder(uncond_input.input_ids.to(device))[0]
        # Об’єднуємо ембеддинги
        embeddings = torch.cat([uncond_embeddings,
text_embeddings])
        # Ініціалізуємо шум
        latents = torch.randn((1, unet.in_channels, height //
8, width // 8), device=device)
        latents = latents * scheduler.init_noise_sigma
        scheduler.set_timesteps(num_inference_steps)
        # Diffusion loop
        for t in scheduler.timesteps:
            # Дублюємо latents для умовного/безумовного CFG
            latent_model_input = torch.cat([latents] * 2)
            # Масштабуємо латенти
            latent_model_input =
scheduler.scale_model_input(latent_model_input, timestep=t)
            # Передбачення шуму
            with torch.no_grad():
```

Продовження лістингу В.1

```
                noise_pred = unet(latent_model_input, t,
encoder_hidden_states=embeddings).sample
            # Розділяємо CFG
            noise_pred_uncond, noise_pred_text =
noise_pred.chunk(2)
            noise_pred = noise_pred_uncond + guidance_scale *
(noise_pred_text - noise_pred_uncond)
            # Крок scheduler'a
            latents = scheduler.step(noise_pred, t,
latents).prev_sample
        # Декодування в зображення
        with torch.no_grad():
            image = vae.decode(latents / 0.18215).sample
        # Перетворення в PIL
        image = (image / 2 + 0.5).clamp(0, 1)
        image = image.cpu().permute(0, 2, 3, 1).numpy()
        image = (image[0] * 255).astype(np.uint8)
        image = Image.fromarray(image)
        return image
```

Лістинг В.2 – Реалізація бізнес–процесу «Корекція зображення»:

```
    # Функція корекції зображення
    def imageChange(self, prompt, image, mask_image,
num_inference_steps=50, guidance_scale=7.5):
        # 1. Токенізація тексту
        text_inputs = self.tokenizer(prompt,
padding="max_length",
max_length=self.tokenizer.model_max_length,
return_tensors="pt")
        text_embeddings =
self.text_encoder(text_inputs.input_ids.to(self.device))[0]
        # 2. Підготовка зображення і маски
```

Продовження лістингу В.2

```
        init_image =
self.image_processor.preprocess(image).to(self.device)
        mask =
self.mask_processor.preprocess(mask_image).to(self.device)
        # 3. Генерація випадкового шуму
        latents = self.prepare_latents(init_image.shape,
dtype=torch.float16, device=self.device)
        timesteps = self.scheduler.timesteps
        # 4. Прогін дифузійного процесу
        for t in timesteps:
            # Додавання шуму
            noisy_latents = self.scheduler.add_noise(latents,
t)
            # Об’єднання з маскою
            latent_input = torch.cat([noisy_latents, mask,
init_image], dim=1)
            # Генерація шуму нейронкою UNet
            noise_pred = self.unet(latent_input, t,
encoder_hidden_states=text_embeddings).sample
            # Крок оновлення latent-змінної
            latents = self.scheduler.step(noise_pred, t,
latents).prev_sample
        # 5. Декодування латентного простору в зображення
        result = self.vae.decode(latents / 0.18215).sample
        image = self.image_processor.postprocess(result)
        return image
```

Лістинг В.3 – Реалізація програмної логіки системи для спілкування користувача з системою:

```
    #Реалізація спілкування користувача з системою
    async def start(update: Update, context:
ContextTypes.DEFAULT_TYPE):
```

Продовження лістингу В.3

```
        await update.message.reply_text("Привіт! Я бот для створення і корекції зображень. Надішли мені опис або фото.")
    async def handle_text(update: Update, context: ContextTypes.DEFAULT_TYPE):
        user_id = update.message.from_user.id
        text = update.message.text
        user_sessions[user_id] = {"prompt": text}
        await update.message.reply_text("Запит збережено. Надішли зображення або /generate для створення нового зображення.")
    async def handle_image(update: Update, context: ContextTypes.DEFAULT_TYPE):
        user_id = update.message.from_user.id
        photo = update.message.photo[-1]  # найкраща якість
        file_path = await photo.get_file()
        image_path = f"{user_id}_input.png"
        await file_path.download_to_drive(image_path)
        # Якщо вже є запит – ймовірно, це корекція
        if user_id in user_sessions and "prompt" in user_sessions[user_id]:
            user_sessions[user_id]["image"] = image_path
            await update.message.reply_text("Зображення збережено. Якщо хочеш, надішли маску або /inpaint.")
    async def generate(update: Update, context: ContextTypes.DEFAULT_TYPE):
        user_id = update.message.from_user.id
        prompt = user_sessions.get(user_id, {}).get("prompt", "an astronaut riding a horse")
        await update.message.reply_text("Генерую зображення, зачекай...")
        image = pipe_txt2img(prompt).images[0]
        output_path = f"{user_id}_result.png"
        image.save(output_path)
```

Продовження лістингу В.3

```
        await
update.message.reply_photo(photo=InputFile(output_path))
    async def inpaint(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        user_id = update.message.from_user.id
        session = user_sessions.get(user_id, {})
        if "prompt" not in session or "image" not in session or
"mask" not in session:
            await update.message.reply_text("Потрібно мати
текст, зображення і маску.")
            return
        await update.message.reply_text("Виконую корекцію...")
        from PIL import Image
        init_image =
Image.open(session["image"]).convert("RGB").resize((1024,
1024))
        mask_image =
Image.open(session["mask"]).convert("RGB").resize((1024, 1024))
        prompt = session["prompt"]
        image = pipe_inpaint(prompt=prompt, image=init_image,
mask_image=mask_image).images[0]
        output_path = f"{user_id}_corrected.png"
        image.save(output_path)
        await
update.message.reply_photo(photo=InputFile(output_path))
```

Лістинг В.4 – Реалізація програмної логіки системи для обробки запитів користувача:

```
    async def start(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        reply_keyboard = [['Generation', 'Correction']]
        context.user_data.clear()
        global remaking
```

Продовження лістингу В.4

```
            remaking = False
            global correction
            correction = False
            await update.message.reply_text(
                '<b>Welcome to the Image Generator Bot!\n'
                'Let\'s get some details about image you\'re
want.\n'
                'What you want to do?</b>',
                parse_mode='HTML',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard,
one_time_keyboard=True, resize_keyboard=True),)
            return MODEL
        async def modelChoose(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
            text = update.message.text
            print(text)
            if text == "Generation":
                user = update.message.from_user
                context.user_data['modelChoose'] =
update.message.text
                reply_keyboard = [['StableDiffusion', 'Runway']]
                context.user_data.clear()
                logger.info('Car type of %s: %s', user.first_name,
update.message.text)
                actions = {"Generation": "🚗", "Correction":
"🚙"}
                await update.message.reply_text(
                    f'<b>You selected {update.message.text} action
{actions[update.message.text]}.\n'
                    f'What model you  will use?</b>',
                    parse_mode='HTML',

reply_markup=ReplyKeyboardMarkup(reply_keyboard,
one_time_keyboard=True, resize_keyboard=True),)
```

Продовження лістингу В.4

```
            return QUERRY
        elif text == "Correction":
            global correction
            correction=True
            user = update.message.from_user
            context.user_data.clear()
            logger.info('Car type of %s: %s', user.first_name,
update.message.text)
            actions = {"Correction": "🚗"}
            await update.message.reply_text(
                f'Send your image for
{update.message.text}.\n',
                parse_mode='HTML',
                reply_markup=ReplyKeyboardRemove(),)
            return DOWNLOADIMAGE
    async def download(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        photo = update.message.photo[-1]
        file = await context.bot.get_file(photo.file_id)
        os.makedirs("downloads", exist_ok=True)
        file_path = f"ai_generated_image.png"
        # Завантажити
        await file.download_to_drive(file_path)
        context.user_data["user_image_path"] = file_path
        await update.message.reply_text("✅ Зображення
збережено!")
        return await Writequerry(update,context)
    async def Writequerry(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        print("wtfffffffffff")
        global correction
        if correction== False:
            user = update.message.from_user
```

Продовження лістингу В.4

```
            context.user_data['modelChoose'] =
update.message.text
            context.user_data.clear()
            logger.info('Car type of %s: %s', user.first_name,
update.message.text)
            actions = {"StableDiffusion": "🚗", "Runway":
"🚙"}
            await update.message.reply_text(
                f'<b>You selected {update.message.text} model
{actions[update.message.text]}.\n'
                f'Write down your prompt</b>',
                parse_mode='HTML',
                reply_markup=ReplyKeyboardRemove(), )
            return RESULT
        else:
            user = update.message.from_user
            context.user_data.clear()
            logger.info('Car type of %s: %s', user.first_name,
update.message.text)
            actions = {"Correction": "🚗"}
            await update.message.reply_text(
                f'Write your prompt for
{update.message.text}.\n',
                parse_mode='HTML',
                reply_markup=ReplyKeyboardRemove(),)
            return CORRECTIMAGE
    def generation(prompt):
        # Генерація зображення
        image = pipe(prompt, guidance_scale=7.5).images[0]
        # Збереження або відображення
        image.save("ai_generated_image.png")
    async def correction_sdxl(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        global prompt
```

Продовження лістингу В.4

```
        correction_text = update.message.text
        prompt = f"{prompt}, {correction_text}"
        print(prompt)
        context.user_data["last_prompt"] = prompt
        global correction
        correction = True
        await update.message.reply_text("Генерую оновлене
зображення...")
        image =
Image.open("ai_generated_image.png").convert("RGB")
        image = image.resize((1024, 1024))
        output = pipe(prompt=prompt, image=image,
strength=0.01, guidance_scale=20)
        updated_image = output.images[0]
        updated_image.save("ai_generated_image.png")
        return await summary(update,context)
    async def summary(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        """Summarizes the user's selections and ends the
conversation, including the uploaded image."""
        global prompt
        global correction
        if correction== False:
            if remaking == False:
                prompt = update.message.text
            generation(prompt)
        print(prompt)
        # Construct the summary text
        img = open('ai_generated_image.png', 'rb')
        await context.bot.send_photo(
            chat_id=update.effective_chat.id,
            photo=img,
            caption='Here your image!')
        chat_id = update.effective_chat.id
```

Продовження лістингу В.4

```
        actions = {'Save Image', 'Regenerate Image','Image
Correction','To Start'}
        reply_keyboard = [['Save Image', 'Regenerate
Image','Image Correction','To Start']]
        reply_markup = ReplyKeyboardMarkup(reply_keyboard,
resize_keyboard=True, one_time_keyboard=True)
        await update.message.reply_text("Choose action:",
reply_markup=reply_markup)
        correction=False
        return CORRECTION
    async def CorrectOrSave(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        text = update.message.text
        print(text)
        if text == "Save Image":
            return await save_photo(update,context)
        elif text == "Regenerate Image":
            global remaking
            remaking=True
            await update.message.reply_text("Regenerating..",
reply_markup=ReplyKeyboardRemove())
            return await summary(update,context)
        elif text == "Image Correction":
            global correction
            correction=True
            print("correction")
            return await Writequerry(update,context)
        elif text == "To Start":
            return await start(update,context)
        else:        await update.message.reply_text("Будь
ласка, оберіть одну з кнопок.")        return END
    async def save_photo(update: Update, context:
ContextTypes.DEFAULT_TYPE):
        with open('ai_generated_image.png', 'rb') as photo:
```

Продовження лістингу В.4

```
            os.makedirs("downloads", exist_ok=True)
            file_path =
f"downloads/photo_{update.message.from_user.id}.jpg"
            with open(file_path, 'wb') as out_file:
                out_file.write(photo.read())
        with open('ai_generated_image.png', 'rb') as f:
            await update.message.reply_document(f,
filename="image.png")
        return await start(update,context)
    async def cancel(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
        """Cancels and ends the conversation."""
        await update.message.reply_text('Bye! Hope to talk to
you again soon.', reply_markup=ReplyKeyboardRemove())
        return ConversationHandler.END)
```

ДОДАТОК Г

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	109с.
	Інші документи	
2.	Презентаційні матеріали	27 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Левицький К.Ю.			Створення та корекція зображення за допомогою текстових підказок у діалозі
Перевірив		Терзіян В. Я.			
Н.контр.		Малєєва І. А.			
Затв.		Золотухін О.В.			

Шифр групи	Код напр./спец.
СШІм-23-1	122
ХНУРЕ кафедра ШІ	