МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Веб програм хөгжүүлэлт
(Full-stack web development)

Програм хангамж(D061302)
Үйлдвэрийн дадлагын тайлан

Удирдагч: _____ Д. Эрдэнэбаяр

Хамтран удирдагч: _____ Н. Оюун-Эрдэнэ

Гүйцэтгэсэн: _____ Д. Балжинням (20B1NUM0563)

Улаанбаатар

2023 оны 9 сар

Зохиогчийн баталгаа

Миний бие Даянгийн Балжинням ”Веб програм хөгжүүлэлт” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг тайлангийн ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

ЗУРГИЙН ЖАГСААЛТ

ХҮСНЭГТИЙН ЖАГСААЛТ

Кодын жагсаалт

УДИРТГАЛ

Миний бие Даянгийн Балжинням ”Веб програм хөгжүүлэлт” сэдэвтэй үйлдвэрийн дадлагын ажлыг Dentsu Data Artist Mongol компани дээр гүйцэтгэсэн. Энэхүү үйлдвэрийн дадлагын хүрээнд Python, болон Javascript програмчиллын хэлнүүд дээр түлхүү ажилсан.

Энэхүү дадлагын ажлын хүрээнд Python-гийн FAST-API framework, Javascript-н Vuejs дээр ажилсан билээ. Хийх ажлын гол зорилго нь, гараас хэрэглэгчийн оруулсан текстийг PNG файл болгон **машин сургалтын** аргаар, шошго үүсгэхэд ашиглах.

Table 1: Дадлагын төлөвлөгөө

№	**Гүйцэтгэх ажил**	**Хугацаа**	**Биелэлт**	**Дадлагын удирдагчийн үнэлгээ**
1	Гараас оруулсан текстийн дагуу PNG үүсгэхэд шаардлагтай технологийг судлах	06/07 - 06/09		
2	Front-End дээр үүсгэсэн PNG file-г үзэх хэсгийг хэрэглэгчид хялбар байдлаар хийх	06/09 - 06/15		
3	Back-End дээр динамик байдлаар хэрэглэгчийн оруулсан текстийг Фонтын хэмжээ, өнгө, чимэглэлийн дагуу үүсгэх	06/15 - 06/23		
4	Ашиглаж болохуйц End-Point үүсгэх	06/23 - 06/24		
5	Үүсгэсэн файлыг ашигласан тохиолдолд AWS ашиглах логик хэрэгжүүлэх	06/24 - 06/26		
6	Database дээр шинэ мөр нэмж migration хийх	06/26 - 06/27		

1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА

1.1 Товч танилцуулга

Dentsu Data Artist Mongol нь 2018 оны 6-р сард, Data Artist Inc.-ийн охин компани болж байгуулагдсан. Дэнцү группын гишүүний хувьд дэлхийн өнцөг булан бүрт байгаа группын компаниудад тоон маркетингийн чиглэлээр өгөгдлийн шинжилгээ, AI model, систем хөгжүүлэх үйлчилгээг үзүүлдэг. Дэнцү грүпп нь маркетингийн чиглэлээр дэлхийд тавд эрэмбэлэгддэг.

1.2 Ямар үйлчилгээ үзүүлдэг вэ?

Уг компани нь мэдээллийн технологийн чиглэлээр үйлчилгээ явуулдаг бөгөөд голчлон Японы компаниудад хиймэл оюун, ухаан машин сургалтын үйлчилгээ үзүүлдэг. Мөн шаардлагатай тохиолдолд систем хөгжүүлэлтийг хийдэг ба тухайн багт нь би байдаг.

2. СИСТЕМИЙН ШААРДЛАГА

2.1 Танилцуулга

Миний дадлагын хугацаанд ажилсан систем нь хиймэл оюун ухаан, машин сургалтын технологийг ашиглан бүх төрлийн шошгыг хэрэглэгчийн оруулсан мэдээллийг ашиглан үүсгэдэг систем юм. Ингэснээр дизайнер хүмүүсийн ажлыг хөнгөвчилж байгаа билээ. Энэхүү систем нь recommendation model ашиглан ямар төрлийн хүмүүст зориулснаар нь ялгаж өөр төрлийн хэв маягийн өнгө, зураг, дизайн сонгодог ба тэрхүү сонгодогсон матералуудыг LAYOUT-GAN++ гэх model ашиглан layout-г нь тохируулж эцсийн бүтээгдэхүүнийг үүсгэдэг.

2.2 Функционал шаардлагууд

- **Хэрэглэгчийн бүртгэл**: Вэбсайт нь хэрэглэгчдэд цахим шуудангийн хаягаа ашиглан бүртгүүлэх, бүртгэл үүсгэх боломжийг олгох ёстой.
- **Хэрэглэгчийн нэвтрэлт**: Бүртгэгдсэн хэрэглэгчид бүртгүүлсэн цахим шуудан болон нууц үгээ ашиглан бүртгэлдээ нэвтрэх боломжтой байх ёстой.
- **Хэрэглэгчийн хяналтын самбар**: Нэвтэрсэний дараа хэрэглэгч өөрийн профайлыг удирдах, шошго үүсгэх функцэд хандах боломжтой хяналтын самбартай байх ёстой.
- **Зураг байршуулах**: Хэрэглэгчид шошго үүсгэхийг хүссэн зургаа байршуулах боломжтой байх ёстой (adobe illustrator file ашиглах ёстой).
- **Машин сургалтын model ашиглах endpoint**: Вэбсайт нь байршуулсан зураг дээр үндэслэн шошго үүсгэх боломжтой endpoint-уудтай байх ёстой.
- **Шошго үүсгэх**: Зургийг байршуулсны дараа вэбсайт нь машин сургалтын model ашиглан шошго үүсгэх ёстой.

- **Шошгоны дэлгэц**: Вэбсайт нь үүсгэсэн шошгыг хэрэглэгчдэд харуулах ёстой.
- **Шошго татаж авах**: Хэрэглэгчид үүсгэсэн шошгыг тохирох форматаар татаж авах боломжтой байх ёстой (AI эсвэл PNG).
- **Хэрэглэгчийн санал хүсэлт**: Хэрэглэгчид үүсгэсэн шошгоны нарийвчлалын талаар санал хүсэлт өгөх боломжтой байх ёстой.

2.3 Функционал бус шаардлагууд

- **Хурд**: Шошго үүсгэх үйл явц нь хамгийн бага хоцрогдолтой, хурдан бөгөөд үр дүнтэй байх ёстой.
- **Аюулгүй байдал**: Хэрэглэгчийн мэдээлэл, үүнд байршуулсан зураг, хамгаалагдсан байх ёстой.
- **Өргөтгөх чадвар (Scalibility)**: Вэбсайт нь хурд алдагдуулахгүйгээр олон тооны хэрэглэгчдэд үйлчлэх чадвартай байх ёстой.
- **Ашиглах боломж**: Вэбсайт нь хэрэглэгчдэд ээлтэй интерфэйстэй, ойлгомжтой зааварчилгаа, хялбар навигацтай байх ёстой.
- **Найдвартай байдал**: Шошго үүсгэхэд ашигладаг машин сургалтын model нь үнэн зөв, найдвартай үр дүнг өгөх ёстой.
- **Хүртээмжтэй байдал**: Вэбсайт нь өөр өөр хөтөч (Chrome, Firefox, Safari гэх мэт) болон төхөөрөмжүүдтэй (ширээний компьютер, гар утас, таблет) нийцтэй байх ёстой.
- **Maintainibility**: Вэбсайт болон түүний үндсэн код нь засвар үйлчилгээ хийх, шинэчлэхэд хялбар байх ёстой.
- **Дагаж мөрдөх**: Вэбсайт нь өгөгдөл хамгаалах хууль, дүрэм журамд нийцсэн байх ёстой.

3. АШИГЛАХ ТЕХНОЛОГИУД

3.1 Back-end талын технологиуд

3.1.1 Python, FastAPI

FastAPI нь python хэлний ASGI[1] framework ба үзүүлэлтийн хувьд nodejs эсвэл go зэргийн үзүүлдэг маш өндөр үзүүлэлтрүү дөхдөг билээ.

- Asynchronous байдлаар ажиллаж чаддаг байдал нь cpu-ны олон цөмийг ашиглах боломжийг олгодог ингэснээр илүү олон хандалт зэрэг авч чадна.
- Pydantic, Starlette гэсэн хоёр сан дээр суурилсан ба, starlette нь ASGI байдлаар ажиллах боломжийг олгох бол, Pydantic нь server дээр validation хийх боломжийг олгодог.
- Database migration хийхэд sqlalchemy ашигладаг ба энэ нь python хэлний ORM[2] ба давуу тал нь хөгжүүлэгч шууд database-тай харьцах биш python-г ашиглан харьцах боломжийг олгоно, ингэснээр database-н схемыг өөрчлөхэд хялбар болохоос гадна database injection зэргээс сэргийлэх давуу талтай.

3.2 Front-end талын технологиуд

3.2.1 Vuejs

Front-end талын хэсгийн технологи бол Vuejs-н progressive framework **Nuxt.js**[3] ба давуу тал нь SSR хийх боломжийг олгодгоос гадна бусад routing, local storage, гэх мэт хөгжүүлжэгчдийн өөрсдөө тохируулдаг зүйлсийг цаанаас нь шийдэж өгсөн байдаг.

[1] Asynchronous Server Gateway Interface

[2] Object Relational Mapping

[3] `https://nextjs.org/`

3.3 Бусад

3.3.1 Amazon S3

Amazon S3 нь Amazon-н cloud service ба энэ нь хэрэглэгчдийн өгөгдөл, зураг, видео, гэх мэт өгөгдөл хадгалах, хэрэглэгчдийн хандахад хялбар байдлаар хандах боломжийг олгодог.

3.3.2 ImageMagick

ImageMagick нь код бичих замаар зурагт засвар оруулдаг сан.

3.3.3 LAYOUTGAN++

LAYOUTGAN++ дээр fine-tune хийснээр, аль болох хэрэглэгчид таалагдахуйц байдлаар шошгон дээрх матералуудыг байршуулах боломжтой болно.

3.3.4 Dockerizing

Орчин үеийн нэгэн гайхалтай технологи бол контайнерчлах юм. Яагаад Docker чухал вэ гэвэл, ямар нэгэн систем хөгжүүлэгчийн компьютер аль эсвэл ямар сервер дээр ажиллаж байгаагаас үл хамааран програм нь өөрийн тусдаа орчинд ажиллах юм. Яг л Virtual machine шиг гэхдээ давуу тал нь Docker host system-ийнхээ цөмийг (kernel)-г ашигладаг учраас маш бага хэмжээгий зай, нөөц ашигладаг.

3.3.5 CI/CD

Мөн сүүлийн үед маш их өргөн түгж байгаа ойлголт бол Continius Integration/Continius Development. Энэ нь програм хангамж ямар ч нөхцөлд хөгжүүлэлт тасралтгүй явж байх орчноор хангадаг ба системд хэзээ ч тасалдал үүсгэхгүй мөн хүний оролцоог маш бага байлгах давуу талтай.

4. ХЭРЭГЖҮҮЛЭЛТ

4.1 Pillow сан ашиглан динамик байдлаар текстээс зураг үүсгэх

4.1.1 Database дээр зурагний мэдээлэл хадгалах table нэмэх

Database-н table дээр өөрчлөлт оруулахдаа бид sqlalchemy ашиглаж байгаа ба доор бичсэн моделийн дагуу бид migrate хийх юм. Үүний тулд back-end ажиллаж байгаа Docker container-лүү shell нээж төслийн root хэсгээс

```
alembic revision --autogenerate -m "your commit message"
```

гэсэн коммандын ашиглан өөрлөлт оруулах мэдээллиг үүсгэн. Дараа нь

```
alembic upgrade head
```

комманд хийснээр Датабазын модел бүрэн өөрчлөгдөнө.

```
class User(Base):
  # ...
  # Other table information

  text_title_string = Column(String(255), nullable=False, default="")
  text_title_color = Column(String(255), nullable=False, default="
     #000000")
  text_title_font_size = Column(Integer, nullable=False, default=0)
  text_title_font_name = Column(String(255), nullable=False, default=
     "")
  text_title_is_vertical = Column(Integer, nullable=False, default=0)
```

Код 4.1: Table-рүү оруулсан өөрчлөлт

4.1.2 API үүсгэх

Back-end дээр API үүсгэхэд анхаарах хэдэн зүйл бий.

1. Validation хийх ингэснээр хэрэглэгчээс ирсэн мэдээллийг шалгаж нэг ёсондоо ажиллахад ямар нэгэн асуудалгүй болгож байгаа юм. Хэрэв буруу мэдээлэл хүсэлт маягаар ирвэл **422 Unprocessable Entity** гэсэн хариу буцаана.

2. Dependency injection байдлаар user-н token-г шалгана ингэснээр хэрэглэгчийн нэвтрэлтийг шалгаж байгаа юм. Хэрэв хэрэглэгч нэвтрээгүй бол **401 Unauthorized**

```
from fastapi import APIRouter, Depends, status
from sqlalchemy.orm import Session
from app import schemas
from app.feature.pillow import pillow_text_generator, preview_text
from app.models.user import User
from app.v1 import deps
router = APIRouter()


@router.post("/", status_code=status.HTTP_201_CREATED)
def pillow_text(
    *,
    obj_in: schemas.PillowBase,
    db: Session = Depends(deps.get_db),
    _: User = Depends(deps.get_current_user),
):
    return preview_text(
        text=obj_in.text,
        is_vertical=obj_in.is_vertical,
        color=obj_in.color,
```

```
            font_size=obj_in.font_size,
            strokewidth=obj_in.stroke_width,
            bordered=obj_in.bordered,
            PPI=obj_in.PPI,
        )
```

Код 4.2: Зураг буцаан user-лүү илгээх endpoint

4.1.3 Үүсгэсэн API дуудах

Энэхүү endpoint-руу user үүсгэх зурагнийхаа текст мэдээллийг JSON хэлбэрээр POST request явуулна. Frontend-с үүсгэсэн Endpointoo ашиглахдаа responseType-г нь arraybuffer болгож байгаа юм ингэснээр base64 image interneteer явуулснаас харьцангуй бага bandwidth ашиглах юм. Үүний дараагаар авсан датагаа decode хийж base64 img болгон хэрэглэгчид харуулна.

```
async preview_text_to_image({ _ }, { data }) {
  const result = await this.$axios.post(`text_image/`, data, {
    responseType: 'arraybuffer',
  })
  if (result.status === 200) {
    const b64 = btoa(String.fromCharCode(...new Uint8Array(result.
       data)))
    const imgData = 'data:' + result.headers['content-type'] + ';
       base64,' + b64
    return imgData
  } else {
    console.log('error')
  }
},
```

Код 4.3: Endpoint дуудах function

4.1.4 Гол script

```
  import io
import os
import random
import typing as t
import uuid

from fastapi.responses import StreamingResponse
from PIL import Image, ImageDraw, ImageFont

from app.core.config import settings
from app.schemas.materials import GeneratedMaterial
from app.utils.s3_handler import upload_file_object

color_dict = {
    "#000000": "#FFFFFF",  # black
    "#FFFFFF": "#000000",  # white
    "#B20019": "#FFFFFF",  # red
    "#270B65": "#FFFFFF",  # blue
    "#158D34": "#FFFFFF",  # green
    "#D38A15": "#FFFFFF",  # orange
    "#F5F105": "#000000",  # yellow
    "#B7AC8B": "#FFFFFF",  # gold
}

```

```
25
26  def get_items_from_child_dir(loc: str):
27      current_directory = os.getcwd()
28      current_directory = f"{current_directory}/{loc}"
29      items = os.listdir(current_directory)
30      return items
31
32
33  FONTS = get_items_from_child_dir("app/feature/fonts")
34
35
36  def vertical_dimensions(*, lines, font, font_size, gaps):
37      lines.reverse()
38      max_char_width = 0
39      for line in lines:
40          for char in line:
41              max_char_width = max(max_char_width, get_text_dimensions(
                    char, font)[2])
42
43      # first max function gets height of every letter and gets the max,
44      # second max function gets the max of the length of each line
45      longest_text = max(len(line) for line in lines)
46      total_line_height = (
47          max(get_text_dimensions(line, font)[3] - get_text_dimensions(
                line, font)[1] for line in lines) * longest_text
48      )
49      padding = 4 * gaps
50      width = (font_size + gaps) * len(lines) + padding
```

```
    height = (font_size + gaps) * longest_text + padding
    return width, height


def horizontal_dimensions(*, lines, font, font_size, gaps):
    padding = 4 * gaps
    width = max(get_text_dimensions(line, font)[2] for line in lines) +
        padding
    height = (font_size + gaps) * len(lines) + padding
    return width, height


def get_text_dimensions(text, font):
    draw = ImageDraw.Draw(Image.new("RGBA", (1, 1), color=(0, 0, 0, 0))
       )
    return draw.textbbox((0, 0), text, font)


def shrink_text(width, height, img, max_width, max_height):
    aspect_ratio = width / height

    if width > max_width or height > max_height:
        if width > height:
            new_width = max_width
            new_height = int(new_width / aspect_ratio)
        else:
            new_height = max_height
            new_width = int(new_height * aspect_ratio)
```

```
77
78          img = img.resize((new_width, new_height), Image.ANTIALIAS)
79      return img
80
81
82  def preview_text(
83      text: str,
84      is_vertical: bool,
85      color: str,
86      bordered: bool,
87      PPI: int,
88      font_size: int,
89      strokewidth: int,
90      gaps=3,
91  ):
92      height = 0
93      width = 0
94      if not bordered:
95          strokewidth = 0
96      current_directory = os.path.dirname(os.path.abspath(__file__)) + "/
           fonts"
97      font_path = os.path.join(current_directory, FONTS[1])
98
99      font = ImageFont.truetype(font_path, size=font_size)
100
101     font_list = get_items_from_child_dir("app/feature/fonts")
102
103     lines = text.split("\n")
```

```
104     if is_vertical:
105         gaps = gaps // 2
106         width, height = vertical_dimensions(
107             lines=lines,
108             font=font,
109             font_size=font_size,
110             gaps=gaps,
111         )
112     else:
113         width, height = horizontal_dimensions(lines=lines, font=font,
                font_size=font_size, gaps=gaps)
114
115     img = Image.new("RGBA", (width, height), color=(0, 0, 0, 0))
116     draw = ImageDraw.Draw(img)
117
118     if is_vertical:
119         write_vertical(
120             lines=lines,
121             draw=draw,
122             font=font,
123             font_size=font_size,
124             gaps=gaps,
125             color=color,
126             outline_color="white",
127             strokewidth=strokewidth,
128         )
129     else:
130         write_horizontal(
```

```
            lines=lines,
            draw=draw,
            font=font,
            font_size=font_size,
            gaps=gaps,
            color=color,
            outline_color="white",
            strokewidth=strokewidth,
        )

    img_data = io.BytesIO()
    dpi = PPI / 2.54
    img.save(img_data, format="PNG")
    img_data.seek(0)

    return StreamingResponse(img_data, media_type="image/png")


def write_vertical(*, lines, draw, font, font_size, gaps, color,
   outline_color, strokewidth):
    padding = gaps * 2
    gaps = gaps // 2
    y = gaps
    x = padding
    # char_width = max(get_text_dimensions(char, font)[2] for line in
       lines for char in line)
    for idx, line in enumerate(lines):
```

```
156         box_w = max(get_text_dimensions(x, font)[2] -
               get_text_dimensions(x, font)[0] for x in line)
157         for char_idx, char in enumerate(line):
158             if char == " ":
159                 char = " "
160             alignment_val = (box_w - get_text_dimensions(char, font)
                   [2]) / 2
161             draw.text(
162                 (x + alignment_val, y),
163                 char,
164                 font=font,
165                 fill=color,
166                 stroke_fill=outline_color,
167                 stroke_width=strokewidth,
168             )
169             y += font_size
170         x += font_size + gaps
171         y = padding
172
173
174 def write_horizontal(*, lines, draw, font, font_size, gaps, color,
       outline_color, strokewidth):
175     padding = gaps * 2
176     y = gaps
177     for line in lines:
178         x = padding
179         draw.text(
180             (x, y),
```

```
            line,
            font=font,
            fill=color,
            spacing=gaps,
            stroke_fill=outline_color,
            stroke_width=strokewidth,
        )
        y += font_size + gaps


def pillow_text_generator(
    text: str,
    is_vertical: bool = True,
    bordered: bool = True,
    PPI: int = 300,
    font_size: int = 90,
    strokewidth: int = 3,
    job_number: int = None,
    text_type: str = "text_title",
    gaps=3,
    element=None,
) -> t.List[str]:
    if text_type == "text_title":
        start_index = -23
    else:
        start_index = -63
    result_list = []
    for each_font in FONTS:
```

```
209         for color, outline_color in color_dict.items():
210             if not bordered:
211                 strokewidth = 0
212             current_directory = f"{os.path.dirname(os.path.abspath(
                    __file__))}/fonts"
213             font_location = os.path.join(current_directory, each_font)
214             font = ImageFont.truetype(font_location, size=font_size)
215 
216             lines = text.split("\n")
217             width = height = 0
218             if is_vertical:
219                 width, height = vertical_dimensions(
220                     lines=lines,
221                     font=font,
222                     font_size=font_size,
223                     gaps=gaps,
224                 )
225             else:
226                 width, height = horizontal_dimensions(lines=lines, font
                        =font, font_size=font_size, gaps=gaps)
227 
228             img = Image.new("RGBA", (width, height), color=(0, 0, 0, 0)
                    )
229             draw = ImageDraw.Draw(img)
230 
231             if is_vertical:
232                 write_vertical(
233                     lines=lines,
```

```
234                     draw=draw,
235                     font=font,
236                     font_size=font_size,
237                     gaps=gaps,
238                     color=color,
239                     outline_color=outline_color,
240                     strokewidth=strokewidth,
241                 )
242             else:
243                 write_horizontal(
244                     lines=lines,
245                     draw=draw,
246                     font=font,
247                     font_size=font_size,
248                     gaps=gaps,
249                     color=color,
250                     outline_color=outline_color,
251                     strokewidth=strokewidth,
252                 )
253
254             saving_location = f"/jobs/{job_number}/input/png"
255             text = text.replace("␣", "_")
256             font_name = each_font.replace(".tff", "")
257             file_name = f"{text_type}_created_{color}_{font_name}.png"
258
259             if settings.IS_LOCAL:
260                 directory = f"/static{saving_location}"
261                 os.makedirs(directory, exist_ok=True)
```

```
                saved_location = f"{directory}/{file_name}"
                img.save(saved_location, "PNG", dpi=(PPI, PPI))
                material_file = GeneratedMaterial(
                    id=start_index,
                    png_width=img.width,
                    png_height=img.height,
                    index=job_number,
                    png_file_path=saved_location,
                    job_id=job_number,
                    element_id=element.id,
                    element_name=element.name,
                    element=element,
                    font_name=font_name,
                    color=color,
                )
                result_list.append(material_file)
            else:
                img_data = io.BytesIO()
                img.save(img_data, format="PNG")
                img_data.seek(0)
                saved_location = f"/static{saving_location}/{file_name}
                    "
                upload_file_object(img_data, saved_location[1:])
                material_file = GeneratedMaterial(
                    id=start_index,
                    png_width=img.width,
                    png_height=img.height,
                    index=job_number,
```

```
                    png_file_path=saved_location,
                    job_id=job_number,
                    element_id=element.id,
                    element_name=element.name,
                    element=element,
                    font_name=font_name,
                    color=color,
                )
                result_list.append(material_file)
            start_index -= 1

    return result_list


def randomly_choose_text(generated_materials: t.List[GeneratedMaterial
    ], registered_materials_from_db):
    black = "black"
    other = "other"
    db_item = "db_item"
    material_map = {
        black: [],
        other: [],
    }

    for item in generated_materials:
        if "#000000".lower() in str(item.png_file_path).lower():
            material_map[black].append(item)
        else:
```

```
316             material_map[other].append(item)
317     if registered_materials_from_db:
318         material_map[db_item] = []
319         for each_item in registered_materials_from_db:
320             material_map[db_item].append(each_item)
321     item_list = []
322     item_weight = []
323     if registered_materials_from_db:
324         item_list.extend([black, other, db_item])
325         item_weight.extend([0.7 * 1 / 3, 0.3 * 1 / 3, 1 / 3])
326     else:
327         item_list.extend([black, other])
328         item_weight.extend([0.7, 0.3])
329 
330     chosen_item: GeneratedMaterial = random.choices(item_list, weights=
        item_weight)[0]
331     chosen_value: GeneratedMaterial = random.choice(material_map[
        chosen_item])
332 
333     return chosen_value
334 
335 
336 def get_font_and_color(ids, titles):
337     font, color = None, None
338     for id in ids:
339         for title in titles:
340             if not isinstance(title, GeneratedMaterial):
341                 continue
```

```
342             if id == title.id:
343                 font, color = title.font_name, title.color
344                 break
345         if font is not None:
346             break
347     return font or "", color or ""
```

Код 4.4: Text-ээс PNG зураг үүсгэдэг script

5. ДҮГНЭЛТ

5.1 Дүгнэлтийн хэсэг

5.1.1 fr fr

Bibliography

[1] Inserting Images, Share LaTex, https://www.sharelatex.com/learn/Inserting_Images

[2] Paragraphs and new lines, Share LaTex, https://www.sharelatex.com/learn/Paragraphs_and_new_lines

[3] Bold, italics and underlining, Share LaTex, https://www.sharelatex.com/learn/Bold,_italics_and_underlining

[4] Lists, Share LaTex, https://www.sharelatex.com/learn/Lists

[5] Tables, Share LaTex, https://www.sharelatex.com/learn/Tables

А. ШИНЖИЛГЭЭ ЗОХИОМЖ

Хавсралтын агуулга

В. КОДЫН ХЭРЭГЖҮҮЛЭЛТ