4 IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi sesuai dengan desain sistem yang telah dibuat dalam bab sebelumnya. Implementasi sistem meliputi proses *Fine tuning* dan pembuatan aplikasi *mobile bible chat*

Tabel 4.1

Pemetaan Desain dan Implementasi

Segmen Program	Desain	Keterangan
4.1	Gambar 3.1	Pembuatan Dataset Bunyi Ayat
4.2	Gambar 3.2	Pembuatan Dataset QnA dari BaDeNo Pertanyaan
4.3	Gambar 3.16	*Code* untuk *Fine Tuning* Llama 2
4.4	Gambar 3.16	*Code* untuk *batch generate* jawaban model Llama 2
4.5	Gambar 3.16	*Code* untuk *Fine Tuning* Mistral
4.6	Gambar 3.16	*Code* untuk *batch generate* jawaban model Mistral
4.7	Gambar 3.17	*Code flutter* fitur *login*
4.8	Gambar 3.17	*Code flutter* fitur *register*
4.9	Gambar 3.17	*Code flutter* fitur *forgot password*
4.10	Gambar 3.18	*Code flutter* untuk *main page*
4.11	Gambar 3.20	*Code flutter* fitur *group* pendalaman Alkitab
4.12	Gambar 3.21	*Code flutter* fitur *forum*
4.13	Gambar 3.23	*Code flutter* fitur *feedback*
4.14	Gambar 3.22	*Code flutter* fitur *minigames*
4.15	Gambar 3.19	*Code flutter* fitur *chat with bible bot*
4.16	Gambar 3.22	*Code flutter* fitur *bible quiz*
4.17	Gambar 3.22	*Code flutter* fitur *bible search word*

4.1 Aplikasi Pemrograman

Pada sub bab ini akan dijelas Gambar 3.17+

kan mengenai aplikasi pemrograman yang digunakan dalam membuat *chatbot* serta aplikasi *bible chat*. Aplikasi tersebut adalah :

- *Visual Studio Code* : *Visual Studio Code* digunakan untuk membuat aplikasi *mobile* dengan *framework* Flutter dan bahasa pemrograman Dart.
- *Google Collab* : *Google Collab* digunakan untuk memproses data serta melakukan *fine tuning* terhadap model Llama2 7b dan Mistral 7b *Instruct*.
- *Firebase* : *Firebase* digunakan sebagai *database* untuk menyimpan data *chat*, *user*, *minigames*, *forum* serta *feedback*.

- *Figma* : *Figma* digunakan untuk membuat *User Interface* (UI) sebagai gambaran sebelum membuat aplikasi *mobile*.

4.2 Implementasi Program

Pada sub bab ini akan dijelaskan mengenai implementasi program yang digunakan dalam penelitian ini baik untuk *Fine tuning* maupun aplikasi

4.2.1 Format dataset ayat Alkitab

Segmen Program 4.1 Membuat dataset bunyi ayat Alkitab injil Yohanes

```
import random
!pip install bs4
import pandas as pd
import requests
from bs4 import BeautifulSoup
import csv

#DatasetAyatTunggal
polaPertanyaan = [
    "Apa isi dari <Format>?",
    "Apa yang disampaikan dalam <Format>?",
    "Apa yang tertulis di <Format>?",
    "Apa yang termuat di <Format>?",
    "Bagaimana bunyi <Format>?",
    "Apa yang dikatakan dalam <Format>?",
    "Apa konten dari <Format>",
]
polaJawaban = [
    "Isi <Format> adalah '<Isi>'",
    "Yang disampaikan dalam <Format> adalah '<Isi>'",
    "Tertulis dalam <Format> '<Isi>'",
    "Termuat di <Format> '<Isi>'",
    "Bunyi dari <Format> adalah '<Isi>'",
    "Yang dikatakan dalam <Format> adalah '<Isi>'",
    "Isi konten dari <Format> adalah '<Isi>'",
]
listDatasetPertanyaan = []
listDatasetJawaban = []
listEvalPertanyaan = []
listEvalJawaban = []

def genText(user,response):
  text = "### User : " + user + "\n\n### Assistant : " + response
  return text

def removeHtml(rawHtml):
    soup = BeautifulSoup(rawHtml, 'html.parser')
    clean_text = soup.get_text()
    return clean_text
```

```
def createList(pertanyaan, jawaban):
  list = []
  instruction = "You are the assistant for Bible study. User contains
the questions that will be asked to you and Assistant is your
response/answer. Always answer in Indonesian for every question."
  for index,pertanyaan in enumerate(pertanyaan):
    list.append([genText(pertanyaan,jawaban[index])])
  return list

def createCsv(array, nama):
  array2 = [["text"]]
  for element in array:
    array2.append(element)
  with open(nama, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(array2)

def getAyat(ayat):
    ayat = ayat.lower()
    try:
        global versiAlkitab
        ayats = ayat.split(" ")
        if(len(ayats) > 3):
            ayatAlkitab = ayats[0] + " " + ayats[1] + " " + ayats[2]
            versi = ayats[3]
        elif(len(ayats) == 3):
            ayatAlkitab = ayats[0] + " " + ayats[1]
            versi = ayats[2]
        else:
            ayatAlkitab = ayat
            versi = "ayt"
        if(versi.lower() == "kjv"):
            versi = "av"
        url = //API SABDA
        response = requests.get(url)
        data = response.json()["results"]["su"]
        if(data["type"] == "verse"):
            data = data["data"]["results"]
            key = str(list(data.keys())[0])
            data = data[key]["bible"]
            versi = str(list(data.keys())[0])
            data = data[versi]
            isi = data["text"]
            return removeHtml(isi)
        elif(data["type"] == "passage" or data["type"] == "chapter"):
            data = data["data"]["results"][0]
            versi = data["version"]
            data = data["res"]
            key = list(data.keys())
            isiAyat = ""
            for kunci in key:
```

```
                datas = data[kunci]["texts"]
                isiAyat   +=   "("   +   datas["verse"]   +   ")   "   +
datas["text"] + "\n"
            isiAyat = removeHtml(isiAyat)
            return isiAyat
    except Exception as e:
        return e
for ayat in range(minAyat,maxAyat + 1):
  print("Ayat " + str(ayat))
  variasi = [kitab + " " + str(pasal) + ":" + str(ayat), kitab + "
Pasal " + str(pasal) + " Ayat " + str(ayat), kitab + " " + str(pasal)
+ " Ayat " + str(ayat)]
  randomEval = random.randint(0,2)
  listRandom = []
  for counter1 in range(3):
    randomChoice = random.randint(0,len(polaPertanyaan) - 1)
    while randomChoice in listRandom:
      randomChoice = random.randint(0,len(polaPertanyaan) - 1)
    listRandom.append(randomChoice)
    if(counter1 == randomEval):

listEvalPertanyaan.append(polaPertanyaan[randomChoice].replace("<Form
at>", variasi[counter1]))

listEvalJawaban.append(polaJawaban[randomChoice].replace("<Format>",
variasi[counter1]).replace("<Isi>", getAyat(variasi[0])))
    else:

listDatasetPertanyaan.append(polaPertanyaan[randomChoice].replace("<F
ormat>", variasi[counter1]))

listDatasetJawaban.append(polaJawaban[randomChoice].replace("<Format>
", variasi[counter1]).replace("<Isi>", getAyat(variasi[0])))

for ayat in range(minAyat,maxAyat - 2):
  randomAyat = random.randint(ayat + 1, maxAyat)
  variasi = [kitab + " " + str(pasal) + ":" + str(ayat) + "-" +
str(randomAyat),
             kitab + " Pasal " + str(pasal) + " Ayat " + str(ayat) +
" sampai " + str(randomAyat),
             kitab + " " + str(pasal) + " Ayat " + str(ayat) + "-" +
str(randomAyat)]
  randomEval = random.randint(0,5)
  listRandom = []
  for counter1 in range(3):
    randomChoice = random.randint(0,len(polaPertanyaan) - 1)
    while randomChoice in listRandom:
      randomChoice = random.randint(0,len(polaPertanyaan) - 1)
    listRandom.append(randomChoice)
    if(counter1 == randomEval):

listEvalPertanyaan.append(polaPertanyaan[randomChoice].replace("<Form
```

```
at>", variasi[counter1]))

listEvalJawaban.append(polaJawaban[randomChoice].replace("<Format>",
variasi[counter1]).replace("<Isi>", getAyat(variasi[0])))
    else:

listDatasetPertanyaan.append(polaPertanyaan[randomChoice].replace("<F
ormat>", variasi[counter1]))

listDatasetJawaban.append(polaJawaban[randomChoice].replace("<Format>
", variasi[counter1]).replace("<Isi>", getAyat(variasi[0])))

for number in range(maxAyat):
  randomAwal = random.randint(2, 3)
  if(randomAwal == 2):
    ayat1 = random.randint(minAyat, maxAyat)
    ayat2 = random.randint(minAyat, maxAyat)
    while(ayat2 == ayat1):
      ayat2 = random.randint(minAyat, maxAyat)
    formats = [
        kitab + " pasal " + str(pasal) + " ayat " + str(ayat1) + "
dan " + str(ayat2),
        kitab + " " + str(pasal) + " ayat " + str(ayat1) + " dan " +
str(ayat2),
        kitab + " pasal " + str(pasal) + ":" + str(ayat1) + " dan " +
str(ayat2),
        kitab + " " + str(pasal) + ":" + str(ayat1) + " dan " +
str(ayat2),
        kitab + " pasal " + str(pasal) + " ayat " + str(ayat1) + "
serta " + str(ayat2),
        kitab + " " + str(pasal) + " ayat " + str(ayat1) + " serta "
+ str(ayat2),
        kitab + " pasal " + str(pasal) + ":" + str(ayat1) + " serta "
+ str(ayat2),
        kitab + " " + str(pasal) + ":" + str(ayat1) + " serta " +
str(ayat2)
    ]
    randomArr = []
    randomArr2 = []
    randomEval = random.randint(0, 3)
    for index in range(4):
      randomFormat = random.randint(0, 7)
      randomPola = random.randint(0, 6)
      while randomPola in randomArr2:
        randomPola = random.randint(0, 6)
      while randomFormat in randomArr:
        randomFormat = random.randint(0, 7)
      randomArr2.append(randomPola)
      randomArr.append(randomFormat)
      if(randomEval == index):

listEvalPertanyaan.append(polaPertanyaan[randomPola].replace("<Format
```

```
>", formats[randomFormat]))

listEvalJawaban.append(polaJawaban[randomPola].replace("<Format>",
formats[randomFormat]).replace("<Isi>", "\n" + kitab + " " +
str(pasal) + ":" + str(ayat1) + " " + getAyat(kitab + " " +
str(pasal) + ":" + str(ayat1)) + "\n" + kitab + " " + str(pasal) +
":" + str(ayat2) + " " + getAyat(kitab + " " + str(pasal) + ":" +
str(ayat2))))
      else:

listDatasetPertanyaan.append(polaPertanyaan[randomPola].replace("<For
mat>", formats[randomFormat]))

listDatasetJawaban.append(polaJawaban[randomPola].replace("<Format>",
formats[randomFormat]).replace("<Isi>", "\n" + kitab + " " +
str(pasal) + ":" + str(ayat1) + " " + getAyat(kitab + " " +
str(pasal) + ":" + str(ayat1)) + "\n" + kitab + " " + str(pasal) +
":" + str(ayat2) + " " + getAyat(kitab + " " + str(pasal) + ":" +
str(ayat2))))
  else:
    ayat1 = random.randint(minAyat, maxAyat)
    ayat2 = random.randint(minAyat, maxAyat)
    ayat3 = random.randint(minAyat, maxAyat)
    while(ayat2 == ayat1):
      ayat2 = random.randint(minAyat, maxAyat)
    while(ayat3 == ayat1 or ayat3 == ayat2):
      ayat3 = random.randint(minAyat, maxAyat)
    formats = [
        kitab + " pasal " + str(pasal) + " ayat " + str(ayat1) + ", "
+ str(ayat2) + " dan " + str(ayat3),
        kitab + " " + str(pasal) + " ayat " + str(ayat1) + ", " +
str(ayat2) + " dan " + str(ayat3),
        kitab + " pasal " + str(pasal) + ":" + str(ayat1) + ", " +
str(ayat2) + " dan " + str(ayat3),
        kitab + " " + str(pasal) + ":" + str(ayat1) + ", " +
str(ayat2) + " dan " + str(ayat3),
        kitab + " pasal " + str(pasal) + " ayat " + str(ayat1) + ", "
+ str(ayat2) + " serta " + str(ayat3),
        kitab + " " + str(pasal) + " ayat " + str(ayat1) + ", " +
str(ayat2) + " serta " + str(ayat3),
        kitab + " pasal " + str(pasal) + ":" + str(ayat1) + ", " +
str(ayat2) + " serta " + str(ayat3),
        kitab + " " + str(pasal) + ":" + str(ayat1) + ", " +
str(ayat2) + " serta " + str(ayat3)
    ]
    randomArr = []
    randomArr2 = []
    randomEval = random.randint(0, 3)
    for index in range(4):
      randomFormat = random.randint(0, 7)
      randomPola = random.randint(0, 6)
      while randomPola in randomArr2:
```

```
        randomPola = random.randint(0, 6)
      while randomFormat in randomArr:
        randomFormat = random.randint(0, 7)
      randomArr2.append(randomPola)
      randomArr.append(randomFormat)
      if(randomEval == index):

listEvalPertanyaan.append(polaPertanyaan[randomPola].replace("<Format
>", formats[randomFormat]))

listEvalJawaban.append(polaJawaban[randomPola].replace("<Format>",
formats[randomFormat]).replace("<Isi>",  "\n"  +  kitab  +  "  "  +
str(pasal)  +  ":"  +  str(ayat1)  +  "  "  +  getAyat(kitab  +  "  "  +
str(pasal) + ":" + str(ayat1)) + "\n" + kitab + " " + str(pasal) +
":" + str(ayat2) + " " + getAyat(kitab + " " + str(pasal) + ":" +
str(ayat2)) + "\n" + kitab + " " + str(pasal) + ":" + str(ayat3) + "
" + getAyat(kitab + " " + str(pasal) + ":" + str(ayat3))))
      else:

listDatasetPertanyaan.append(polaPertanyaan[randomPola].replace("<For
mat>", formats[randomFormat]))

listDatasetJawaban.append(polaJawaban[randomPola].replace("<Format>",
formats[randomFormat]).replace("<Isi>",  "\n"  +  kitab  +  "  "  +
str(pasal)  +  ":"  +  str(ayat1)  +  "  "  +  getAyat(kitab  +  "  "  +
str(pasal) + ":" + str(ayat1)) + "\n" + kitab + " " + str(pasal) +
":" + str(ayat2) + " " + getAyat(kitab + " " + str(pasal) + ":" +
str(ayat2)) + "\n" + kitab + " " + str(pasal) + ":" + str(ayat3) + "
" + getAyat(kitab + " " + str(pasal) + ":" + str(ayat3))))

formats = [
    kitab + " " + str(pasal),
    kitab + " pasal " + str(pasal)
]
randomEval = random.randint(0, 3)
randomListArr = []
for i in range(3):
  randomFormat = random.randint(0, 6)
  randomFormats = random.randint(0, 1)
  while randomFormat in randomListArr:
    randomFormat = random.randint(0, 6)
  randomListArr.append(randomFormat)
  if(randomEval == i):

listEvalPertanyaan.append(polaPertanyaan[randomFormat].replace("<Form
at>", formats[randomFormats]))

listEvalJawaban.append(polaJawaban[randomFormat].replace("<Format>",
formats[randomFormats]).replace("<Isi>", getAyat(formats[0])))
  else:

listDatasetPertanyaan.append(polaPertanyaan[randomFormat].replace("<F
```

```
ormat>", formats[randomFormats]))

listDatasetJawaban.append(polaJawaban[randomFormat].replace("<Format>
", formats[randomFormats]).replace("<Isi>", getAyat(formats[0])))

createCsv(createList(listDatasetPertanyaan,listDatasetJawaban),
"YT2.csv")
createCsv(createList(listEvalPertanyaan,listEvalJawaban), "YE2.csv")
```

Code diatas merupakan *code* untuk membuat dataset tentang bunyi dari ayat Alkitab versi Alkitab Yang Terbuka, baik ayat tunggal, ayat *range*, dan 1 pasal Yohanes

4.2.2 Data Augmentation untuk BaDeNo Pertanyaan

Segmen Program 4.2 *Data Augmentation* pada data BaDeNo Pertanyaan dengan ChatGPT API

```
!pip install openai
import time
from openai import OpenAI
import json

def genText(system,user,response):
  text = "### System : " + system + "\n\n### User : " + user +
"\n\n### Assistant : " + response
  return text
import csv
import json

nama_file_csv = '/content/sample_data/Data_BaDeNo_Yohanes.csv'
inputs = []
outputs = []
with open(nama_file_csv, 'r') as file_csv:
    pembaca_csv = csv.DictReader(file_csv)

    for baris in pembaca_csv:
        input = baris['question']
        output = baris['answer']
        inputs.append(input)
        outputs.append(output)
indexu = 0
apiKey = ["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", "sk-
O9cstsSPDDCn90AZbRwHT3BlbkFJnRzAhWIOL0pg1FDUonU7",               "sk-
g8QYOJD4X1oDzNXXblLlT3BlbkFJcPjjiA4S0BJ3b1Qnzv2o",               "sk-
p5v7Dzxd7Rb0objlbOyJT3BlbkFJ9m6d1cR2n7ymAC7pcrWX",               "sk-
EOQ7QYbmMFUOR92JyAGOT3BlbkFJR97zcRwMxEdTLZr4he8a"]
client = OpenAI(api_key=apiKey[indexu])

def genPertanyaan(input):
  response = client.chat.completions.create(
    model="gpt-3.5-turbo-1106",
    response_format={ "type": "json_object" },
    messages=[
      {"role": "system", "content": "Buatlah 4 variasi pertanyaan
```

```
dari pertanyaan yang saya berikan dan jangan ubah makna, inti dan
konteks dari pertanyaan yang dibuat variasinya tersebut.Jawab dalam
bahasa Indonesia. kunci JSONnya adalah 'pertanyaan1','pertanyaan2'
sampai 'pertanyaan8'"},
      {"role": "user", "content": input}
    ]
  )
  time.sleep(20)
  return response.choices[0].message.content
dataPertanyaan = []
dataJawaban = []
dataPertanyaan2 = []
dataJawaban2 = []
for index,i in enumerate(inputs):
  while True:
    try:
      print("Proses (" + str(index+1) + "/" + str(len(inputs)) + ")")
      jsons = genPertanyaan(i)
      data = json.loads(jsons)
      count = 0
      pertanyaan1 = data["pertanyaan1"]
      pertanyaan2 = data["pertanyaan2"]
      pertanyaan3 = data["pertanyaan3"]
      pertanyaan4 = data["pertanyaan4"]
      for j in range(5):
        dataJawaban.append(outputs[index])
      dataPertanyaan.append(i)
      dataPertanyaan.append(pertanyaan1)
      dataPertanyaan.append(pertanyaan2)
      dataPertanyaan.append(pertanyaan3)
      dataPertanyaan.append(pertanyaan4)
      break
    except Exception as e:
      print(e)
      if(indexu == len(apiKey) - 1):
        indexu = 0
      else:
        indexu += 1
      client = OpenAI(api_key=apiKey[indexu])

instruction = "You are the assistant for Bible study. User contains
the questions that will be asked to you and Assistant is your
response/answer. Always answer in Indonesian for every question."
files = [["question","answer"]]
pertama = 0
for idx, text in enumerate(dataPertanyaan):
  arrays = []
  arrays.append(text)
  arrays.append(dataJawaban[idx])
  files.append(arrays)
```

Code diatas merupakan *code* untuk melakukan *data augmentation* pada data BaDeNo Pertanyaan dengan menggunakan ChatGPT API.

4.2.3 Fine tuning Llama2 7b

Segmen Program 4.3 *Code Fine Tuning* Llama 2

```
!pip install -q accelerate==0.21.0 peft==0.4.0 bitsandbytes==0.40.2
transformers==4.31.0 trl==0.4.7
!pip install numpy
import numpy as np
import random
import os
import torch
from datasets import load_dataset
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    BitsAndBytesConfig,
    HfArgumentParser,
    TrainingArguments,
    pipeline,
    logging,
)
from peft import LoraConfig, PeftModel
from trl import SFTTrainer

import pandas as pd
from datasets import Dataset
df = pd.read_csv('/content/sample_data/DTYohanes(6).csv')
df = df.sample(frac=1, random_state=42)
dataset = Dataset.from_pandas(df)
print(dataset)
df = pd.read_csv('/content/sample_data/DTYohanes(6).csv')
df = df.sample(frac=1, random_state=42)[0:int(len(df)/5)]
datasetVal = Dataset.from_pandas(df)
print(datasetVal)

model_name = "NousResearch/Llama-2-7b-chat-hf"
new_model = "llama-2-bible"
device_map = {"": 0}
compute_dtype = getattr(torch, "float16")
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=compute_dtype,
    bnb_4bit_use_double_quant=False,
)
tokenizer = AutoTokenizer.from_pretrained(model_name,
trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    device_map=device_map
)
model.config.use_cache = False
model.config.pretraining_tp = 1

peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",
)

training_arguments = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
    optim=optim,
    save_steps=save_steps,
    logging_steps=logging_steps,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    fp16=fp16,
    bf16=bf16,
    max_grad_norm=max_grad_norm,
    max_steps=max_steps,
    warmup_ratio=warmup_ratio,
    group_by_length=group_by_length,
    lr_scheduler_type=lr_scheduler_type,
    report_to="tensorboard"
)

peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_r,
    bias="none",
    task_type="CAUSAL_LM",
)

training_arguments = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=num_train_epochs,
    per_device_train_batch_size=per_device_train_batch_size,
    gradient_accumulation_steps=gradient_accumulation_steps,
    optim=optim,
    save_steps=save_steps,
```

```
    logging_steps=logging_steps,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    fp16=fp16,
    bf16=bf16,
    max_grad_norm=max_grad_norm,
    max_steps=max_steps,
    warmup_ratio=warmup_ratio,
    group_by_length=group_by_length,
    lr_scheduler_type=lr_scheduler_type,
    report_to="tensorboard"
)

trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    eval_dataset=datasetVal,
    peft_config=peft_params,
    dataset_text_field="text",
    max_seq_length=None,
    tokenizer=tokenizer,
    args=training_params,
    packing = False,
)
trainer.train()
```

Code diatas merupakan *code* untuk melakukan proses *fine tuning* pada model Llama 2 menggunakan metode *Supervized Fine Tuning*.

Segmen Program 4.4 *Code generate* hasil jawaban *batch*

```
namaFileInput = "/content/sample_data/pertanyaan_testing.txt"
namaFileOutput   =   "/content/drive/MyDrive/Jawaban_Llama-5e5-2epoch-
batch4.txt"

def writeFile(namaFile,content):
      with open(namaFile, 'a') as file:
            file.write(content + '\n')

def readFile(file_name):
    try:
        with open(file_name, 'r') as file:
            content = file.read()
        return content
    except FileNotFoundError:
        return f"File '{file_name}' tidak ditemukan."

pertanyaan = readFile(namaFileInput).split("\n")

import time
logging.set_verbosity(logging.CRITICAL)
for user in pertanyaan:
  start_time = time.time()
```

```
  instruksi = "You are the assistant for Bible study. User contains the questions that will be asked to you and Assistant is your response/answer. Always answer in Indonesian for every question."
  prompt = "### Instruction : " + instruksi + "\n\n\n### User : " + user + "\n\n\n### Assistant : "
  pipe = pipeline(task="text-generation", model=model, tokenizer=tokenizer, max_length=300 + (len(user) * 2), temperature = 0.1)
  result = pipe(prompt)
  result = result[0]['generated_text']
  jawaban = result.split("\n\n\n")[2].replace("### Assistant : ","")
  pertanyaan = result.split("\n\n\n")[1].replace("### User : ","")
  end_time = time.time()
  print("Pertanyaan : " + pertanyaan + "\nJawaban : " + jawaban)
  print("Elapsed Time : " + str(int(end_time - start_time)))
```

Code diatas merupakan *code* untuk *testing model* dengan berbagai pertanyaan yang telah disiapkan secara *batch*. Dengan *input file* dalam *format* txt serta *output file* dalam *format* txt.

4.2.4 *Fine tuning* Mistral 7b *Instruct*

Segmen Program 4.5 *Code Fine Tuning* Mistral

```
!git clone 'https://github.com/ali7919/Enlighten-Instruct.git'
!pip install -U bitsandbytes
!pip install transformers==4.36.2
!pip install -U peft
!pip install -U accelerate
!pip install -U trl
!pip install datasets==2.16.0
!pip install sentencepiece
from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig,HfArgumentParser,TrainingArguments,pipeline, logging
from peft import LoraConfig, PeftModel, prepare_model_for_kbit_training, get_peft_model
import os,torch
from datasets import load_dataset
from trl import SFTTrainer
import pandas as pd
import pyarrow as pa
import pyarrow.dataset as ds
import pandas as pd
from datasets import Dataset
import re

import pandas as pd
from datasets import Dataset
df = pd.read_csv('/content/drive/MyDrive/DTYohanes(2).csv')
df = df.sample(frac=1, random_state=42)
dataset = Dataset.from_pandas(df)
panjang = len(df)
```

```
print(dataset)
df = pd.read_csv('/content/drive/MyDrive/DEYohanes(2).csv')
df = df.sample(frac=1, random_state=42)
datasetVal = Dataset.from_pandas(df)
print(datasetVal)

base_model = "mistralai/Mistral-7B-Instruct-v0.2"
new_model = "Mistral_Bible"
bnb_config = BitsAndBytesConfig(
    load_in_4bit= True,
    bnb_4bit_quant_type= "nf4",
    bnb_4bit_compute_dtype= torch.bfloat16,
    bnb_4bit_use_double_quant= False,
)
model = AutoModelForCausalLM.from_pretrained(
        base_model,
        load_in_4bit=True,
        quantization_config=bnb_config,
        torch_dtype=torch.bfloat16,
        device_map="auto",
        trust_remote_code=True,
model = prepare_model_for_kbit_training(model)
peft_config = LoraConfig(
    lora_alpha=64,
    lora_dropout=0.1,
    r=64,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=["q_proj",           "k_proj",           "v_proj",
"o_proj","gate_proj"]
)
model = get_peft_model(model, peft_config)
peft_params = LoraConfig(
    lora_alpha=64,
    lora_dropout=0.1,
    r=64,
    bias="none",
    task_type="CAUSAL_LM",
)
training_params = TrainingArguments(
    per_device_train_batch_size=2,
    gradient_accumulation_steps=2,
    optim="paged_adamw_32bit",
    logging_steps=1,
    learning_rate=1e-4,
    fp16=True,
    max_grad_norm=0.3,
    output_dir="./mistral",
    num_train_epochs=3,
    evaluation_strategy="steps",
    eval_steps=0.2,
    warmup_ratio=0.03,
```

```
    save_strategy="epoch",
    group_by_length=True,
    report_to="tensorboard",
    save_safetensors=True,
    lr_scheduler_type="constant",
    seed=42,
)
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    eval_dataset=datasetVal,
    peft_config=peft_params,
    dataset_text_field="text",
    max_seq_length=None,
    tokenizer=tokenizer,
    args=training_params,
    packing = False,
)
trainer.train()
```

Code diatas digunakan untuk *training model* Mistral dengan metode *Supervized Fine Tuning* (SFT) dengan *parameter, data* dan *tokenizer* yang telah ditentukan.

Segmen Program 4.6 *Code generate* hasil jawaban *batch*

```
namaFileInput = "/content/sample_data/pertanyaan_testing.txt"
namaFileOutput = "/content/drive/MyDrive/jawaban_testing3.txt"

def writeFile(namaFile,content):
      with open(namaFile, 'a') as file:
            file.write(content + '\n')

def readFile(file_name):
    try:
        with open(file_name, 'r') as file:
            content = file.read()
        return content
    except FileNotFoundError:
        return f"File '{file_name}' tidak ditemukan."

pertanyaan = readFile(namaFileInput).split("\n")
logging.set_verbosity(logging.CRITICAL)
for user in pertanyaan:
  instruksi = "You are the assistant for Bible study. User contains the questions that will be asked to you and Assistant is your response/answer. Always answer in Indonesian for every question."
  prompt = "### Instruction : " + instruksi + "\n\n### User : " + user + "\n\n### Assistant : "
  pipe = pipeline(task="text-generation", model=model, tokenizer=tokenizer, max_length=(200 + int(len(user) * 2)), temperature = 0.1)
  result = pipe(prompt)
```

```
  result = result[0]['generated_text']
  jawaban = result.split("\n\n")[2].replace("### Assistant : ","")
  pertanyaan = result.split("\n\n")[1].replace("### User : ","")
  writeFile(namaFileOutput,"Pertanyaan : " + pertanyaan + "\nJawaban
: " + jawaban)
```

Code diatas merupakan *code* untuk *testing model* dengan berbagai pertanyaan yang telah disiapkan secara *batch*. Dengan *input file* dalam *format* txt serta *output file* dalam *format* txt.

4.2.5 Login

Pada segmen *login*, *user* dapat memasukkan *username* dan *password* apabila *user* sudah memiliki akun. Apabila *user* belum memiliki akun, *user* dapat menekan tombol *sign up* untuk membuat akun. Apabila *user* lupa password maka *user* dapat menekan tombol *forgot password*.

Segmen Program 4.7 *Code flutter* fitur *login*

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'landing_page.dart';
import 'function.dart';
import 'user.dart';
import 'forgot_pass.dart';
import 'register.dart';

class Login extends StatefulWidget {
  @override
  _LoginState createState() => _LoginState();
}

class _LoginState extends State<Login> {
  final TextEditingController _uname = TextEditingController();
  final TextEditingController _pass = TextEditingController();
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  String _errorMessage = '';

  void _login() async {
    String username = _uname.text;
    String password = _pass.text;

    bool canLogin = await authenticateUser(username, password);
    print(canLogin);
    if (canLogin == true) {
      context.read<UserModel>().setUsername(username);
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => LandingPage()),
      );
```

```
    } else {
      setState(() {
        _errorMessage = 'Username atau Password salah';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        resizeToAvoidBottomInset: false,
        body: SingleChildScrollView(
          padding:                                EdgeInsets.only(bottom:
MediaQuery.of(context).viewInsets.bottom),
          child: Form(
            key: _formKey,
            child: Container(
              width: double.maxFinite,
              padding:  const  EdgeInsets.only(left:  27,  top:  94,
right: 27),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Center(
                    child: Text(
                      "Bible Chat",
                      style: Theme.of(context).textTheme.bodyLarge,
                    ),
                  ),
                  const SizedBox(height: 83),
                  Padding(
                    padding: const EdgeInsets.only(left: 7),
                    child: TextFormField(
                      controller: _uname,
                      decoration: const InputDecoration(
                        hintText: "Username",
                      ),
                    ),
                  ),
                  const SizedBox(height: 27),
                  Padding(
                    padding: const EdgeInsets.only(left: 7),
                    child: TextFormField(
                      controller: _pass,
                      decoration: const InputDecoration(
                        hintText: "Password",
                      ),
                      obscureText: true,
                    ),
                  ),
                  const SizedBox(height: 10),
```

```
                    if (_errorMessage.isNotEmpty)
                      Padding(
                        padding: const EdgeInsets.only(left: 7, bottom:
10),
                        child: Text(
                          _errorMessage,
                          style: TextStyle(color: Colors.red),
                        ),
                      ),
                    const SizedBox(height: 7),
                    Padding(
                      padding: const EdgeInsets.only(left: 7),
                      child: GestureDetector(
                        onTap: () {
                          Navigator.push(context,
MaterialPageRoute(builder: (context) => ForgotPasswordPage()),);
                        },
                        child: Text(
                          "Forgot Password",
                          style:
Theme.of(context).textTheme.bodySmall!.copyWith(decoration:
TextDecoration.underline,color: Colors.blue,),
                        ),
                      ),
                    ),
                    const SizedBox(height: 14),
                    Center(
                      child: SizedBox(
                        width: double.infinity,
                        child: ElevatedButton(
                          child: const Text("LOGIN"),
                          onPressed: _login,
                        ),
                      ),
                    ),
                    const SizedBox(height: 16),
                    Row(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        Padding(
                          padding: const EdgeInsets.only(bottom: 1),
                          child: Text(
                            "Don't have an account?",
                            style:
Theme.of(context).textTheme.bodySmall,
                          ),
                        ),
                        GestureDetector(
                          onTap: () {
                            Navigator.push(context,
MaterialPageRoute(builder: (context) => RegisterPage()),);
                          },
```

```
                        child: Padding(
                          padding: const EdgeInsets.only(left: 2),
                          child: Text(
                            "Sign Up",
                            style:
Theme.of(context).textTheme.bodySmall!.copyWith(color: Colors.blue),
                          ),
                        ),
                      ),
                    ],
                  ),
                  const SizedBox(height: 5),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

4.2.6 Register

Pada segmen *register*, *user* dapat membuat akun dengan mengisi *email*,*username* dan *password*.

Segmen Program 4.8 *Code flutter* fitur *register*

```
import 'package:flutter/material.dart';
import 'function.dart';
import 'login.dart';

class RegisterPage extends StatefulWidget {
  @override
  _RegisterPageState createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  final TextEditingController _uname = TextEditingController();
  final TextEditingController _pass = TextEditingController();
  final TextEditingController _email = TextEditingController();
  final TextEditingController _otp = TextEditingController();
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  String _errorMessage = '';
  String otp = "";

  void _register(BuildContext context) async {
    String username = _uname.text;
    String password = _pass.text;
    String email = _email.text;
```

```
    bool akunSudahAda = await checkUsername(username, email);
    if (akunSudahAda) {
      setState(() {
        _errorMessage = 'Username / Email sudah terpakai';
      });
    } else {
      await addUser(username, email, password);
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => Login()),
      );
    }
  }

  Future<void> _getOtp(String email) async {
    String otps = await getOtp(email);
    otp = otps;
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        resizeToAvoidBottomInset: false,
        body: SingleChildScrollView(
          padding:                                EdgeInsets.only(bottom:
MediaQuery.of(context).viewInsets.bottom),
          child: Form(
            key: _formKey,
            child: Container(
              width: double.infinity,
              padding: const EdgeInsets.all(27.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  Text(
                    "Bible Chat",
                    style: Theme.of(context).textTheme.bodyLarge,
                    textAlign: TextAlign.center,
                  ),
                  const SizedBox(height: 83),
                  Padding(
                    padding:  const  EdgeInsets.symmetric(horizontal:
7),
                    child: TextField(
                      controller: _email,
                      decoration:   const   InputDecoration(hintText:
"Email"),
                    ),
                  ),
                  const SizedBox(height: 27),
```

```
                    Padding(
                      padding:  const  EdgeInsets.symmetric(horizontal:
7),
                      child: TextField(
                        controller: _uname,
                        decoration:   const   InputDecoration(hintText:
"Username"),
                      ),
                    ),
                    const SizedBox(height: 27),
                    Padding(
                      padding:  const  EdgeInsets.symmetric(horizontal:
7),
                      child: TextField(
                        controller: _pass,
                        decoration:   const   InputDecoration(hintText:
"Password"),
                        obscureText: true,
                      ),
                    ),
                    const SizedBox(height: 17),
                    Row(
                      children: [
                        Expanded(
                          child: TextField(
                            controller: _otp,
                            decoration: InputDecoration(
                              border: OutlineInputBorder(),
                              labelText: 'Enter OTP',
                            ),
                          ),
                        ),
                        SizedBox(width: 10),
                        ElevatedButton(
                          onPressed: () async {
                            await _getOtp(_email.text);
                            if(otp != ""){
                              setState(() {
                                _errorMessage = 'OTP Terkirim';
                              });
                            }else{
                              setState(() {
                                _errorMessage = 'OTP Gagal Terkirim';
                              });
                            }
                          },
                          child: Text('Get OTP'),
                        ),
                      ],
                    ),
                    const SizedBox(height: 17),
                    if (_errorMessage.isNotEmpty)
```

```
                    Padding(
                      padding: const EdgeInsets.only(left: 7, bottom:
10),
                      child: Text(
                        _errorMessage,
                        style: const TextStyle(color: Colors.red),
                      ),
                    ),
                  ElevatedButton(
                    onPressed: () {
                      if(_otp.text == otp){
                        _register(context);
                      }else{
                        setState(() {
                          _errorMessage = 'OTP Salah';
                        });
                      }
                    },
                    child: const Text("REGISTER"),
                  ),
                  const SizedBox(height: 13),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Padding(
                        padding: const EdgeInsets.only(bottom: 1),
                        child: Text(
                          "Already have account?",
                          style:
Theme.of(context).textTheme.bodySmall,
                        ),
                      ),
                      GestureDetector(
                        onTap: () {

Navigator.push(context,MaterialPageRoute(builder:     (context)     =>
Login()),);
                        },
                        child: Padding(
                          padding: const EdgeInsets.only(left: 2),
                          child: Text(
                            "Log In",
                            style:
Theme.of(context).textTheme.bodySmall!.copyWith(color: Colors.blue),
                          ),
                        ),
                      ),
                    ],
                  ),
                ],
              ),
            ),
```

```
          ),
        ),
      ),
    );
  }
}
```

4.2.7 Forgot Password

Pada segmen *forgot password*, *user* dapat memulihkan *password* dengan cara memasukkan *email*, *username*, *password* dan konfirmasi *password.*

Segmen Program 4.9 *Code flutter* fitur *forgot password*

```
import 'package:flutter/material.dart';
import 'function.dart';
import 'login.dart';

class ForgotPasswordPage extends StatefulWidget {
  @override
  _ForgotPasswordPageState              createState()              =>
_ForgotPasswordPageState();
}

class _ForgotPasswordPageState extends State<ForgotPasswordPage> {
  final TextEditingController _uname = TextEditingController();
  final TextEditingController _email = TextEditingController();
  final TextEditingController _pass = TextEditingController();
  final TextEditingController _otp = TextEditingController();
  final TextEditingController _retypePass = TextEditingController();
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  String _errorMessage = '';
  String otp = "";
  int valid = 0;

  void _changePassword(BuildContext context) async {

    if (_pass.text == _retypePass.text) {
      bool berhasil = await updatePassword(_uname.text, _pass.text);
      if (berhasil) {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => Login()),
        );
      } else {
        setState(() {
          _errorMessage = 'Gagal mengganti password';
        });
      }
    } else {
      setState(() {
```

```
        _errorMessage = 'Password tidak sama';
      });
    }
  }

  Future<void> _verifEmailUser(String email, String uname) async {
    print("masok");
    bool berhasil = await verification(uname, email);
    print(berhasil);
    setState(() {
      valid = 1;
    });
  }

  void _getOtp(String email) async {
    String otps = await getOtp(email);
    otp = otps;
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        resizeToAvoidBottomInset: false,
        body: SingleChildScrollView(
          padding:                                EdgeInsets.only(bottom:
MediaQuery.of(context).viewInsets.bottom),
          child: Form(
            key: _formKey,
            child: Container(
              width: double.maxFinite,
              padding:  const  EdgeInsets.only(left:  27,  top:  94,
right: 27),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Center(
                    child: Text(
                      "Forgot Password",
                      style: Theme.of(context).textTheme.bodyLarge,
                    ),
                  ),
                  const SizedBox(height: 83),
                  Padding(
                    padding: const EdgeInsets.only(left: 7),
                    child: TextFormField(
                      controller: _email,
                      decoration: const InputDecoration(
                        hintText: "Email",
                      ),
                    ),
                  ),
```

```
                    const SizedBox(height: 27),
                    Padding(
                      padding: const EdgeInsets.only(left: 7),
                      child: TextFormField(
                        controller: _uname,
                        decoration: const InputDecoration(
                          hintText: "Username",
                        ),
                      ),
                    ),
                    const SizedBox(height: 27),
                    Padding(
                      padding: const EdgeInsets.only(left: 7),
                      child: TextFormField(
                        controller: _pass,
                        decoration: const InputDecoration(
                          hintText: "New Password",
                        ),
                        obscureText: true,
                      ),
                    ),
                    const SizedBox(height: 27),
                    Padding(
                      padding: const EdgeInsets.only(left: 7),
                      child: TextFormField(
                        controller: _retypePass,
                        decoration: const InputDecoration(
                          hintText: "Retype New Password",
                        ),
                        obscureText: true,
                      ),
                    ),
                    const SizedBox(height: 27),
                    Row(
                      children: [
                        Expanded(
                          child: TextField(
                            controller: _otp,
                            decoration: InputDecoration(
                              border: OutlineInputBorder(),
                              labelText: 'Enter OTP',
                            ),
                          ),
                        ),
                        SizedBox(width: 10),
                        ElevatedButton(
                          onPressed: () async {
                            await          _verifEmailUser(_email.text,
_uname.text);
                            if(valid == 1){
                              _getOtp(_email.text);
                              setState(() {
```

```
                                _errorMessage = 'OTP Terkirim';
                              });
                            }else{
                              setState(() {
                                _errorMessage = 'Username / Email
tidak valid';
                              });
                            }
                          },
                          child: Text('Get OTP'),
                        ),
                      ],
                    ),
                    const SizedBox(height: 10),
                    if (_errorMessage.isNotEmpty)
                      Padding(
                        padding: const EdgeInsets.only(left: 7,
bottom: 10),
                        child: Text(
                          _errorMessage,
                          style: TextStyle(color: Colors.red),
                        ),
                      ),
                    const SizedBox(height: 10),
                    Center(
                      child: SizedBox(
                        width: double.infinity,
                        child: ElevatedButton(
                          onPressed: () {
                            if(otp == _otp.text){
                              _changePassword(context);
                            }
                            else{
                              setState(() {
                                _errorMessage = 'OTP Salah';
                              });
                            }
                          },
                          child: Text("CHANGE PASSWORD"),
                        ),
                      ),
                    ),
                    const SizedBox(height: 13),
                    Row(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        Padding(
                          padding: const EdgeInsets.only(bottom: 1),
                          child: Text(
                            "Remembered your password?",
                            style:
Theme.of(context).textTheme.bodySmall,
```

```
                        ),
                      ),
                      GestureDetector(
                        onTap: () {
                          Navigator.pop(context);
                        },
                        child: Padding(
                          padding: const EdgeInsets.only(left: 2),
                          child: Text(
                            "Log In",
                            style:
Theme.of(context).textTheme.bodySmall!.copyWith(color: Colors.blue),
                          ),
                        ),
                      ),
                    ],
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

4.2.8 Main Page

Pada segmen *main page*, *user* dapat mengakses banyak *menu* seperti *chat* dengan *Bible bot*, grup pendalaman Alkitab, *forum*, *minigames,* dan *feedback*.

Segmen Program 4.10 *Code flutter* untuk *main page*

```
import 'package:flutter/material.dart';
import 'dart:async';
import 'package:provider/provider.dart';
import 'function.dart';
import 'user.dart';
import 'group_chat.dart';
import 'minigames.dart';
import 'feedback.dart';
import 'forum.dart';
import 'chatbot.dart';

class LandingPage extends StatefulWidget {
  @override
  _LandingPageState createState() => _LandingPageState();
}

class _LandingPageState extends State<LandingPage> {
  List<Map<String, String>> groups = [];
```

```
  Future<void> _getList(String username) async {
    List<Map<String,      String>>      groupList      =      await
getUserGroupList(username);
    setState(() {
      groups = groupList;
    });
  }

  @override
  Widget build(BuildContext context) {
    String username = context.watch<UserModel>().username;
    return SafeArea(
      child: Scaffold(
        body: Container(
          width: double.maxFinite,
          padding: EdgeInsets.symmetric(horizontal: 32),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                "Bible Chat",
                style:
Theme.of(context).textTheme.bodyLarge?.copyWith(
                      fontWeight: FontWeight.bold,
                    ),
              ),
              SizedBox(height: 48),
              Container(
                height: 107,
                width: 107,
                decoration: BoxDecoration(
                  color: Theme.of(context).colorScheme.primary,
                  borderRadius: BorderRadius.circular(53),
                ),
                child: Icon(
                  Icons.person,
                  size: 60,
                  color: Colors.white,
                ),
              ),
              SizedBox(height: 16),
              Text(
                username,
                style: Theme.of(context).textTheme.bodyLarge,
              ),
              SizedBox(height: 42),
              _buildButton(context, 'Chat With Bible Bot', () {
                Navigator.push(
                    context, MaterialPageRoute(builder: (context) =>
Chatbot()));
              }),
```

```
              _buildButton(context, 'Group Pendalaman Alkitab', () async {
                await _getList(username);
                Navigator.push(
                  context,
                  MaterialPageRoute(builder:      (context)      => GroupListPage(groups: groups)),
                );
              }),
              _buildButton(context, 'Forum', () {
                Navigator.push(
                    context, MaterialPageRoute(builder: (context) => ForumPage()));
              }),
              _buildButton(context, 'Minigames', () {
                Navigator.push(
                    context, MaterialPageRoute(builder: (context) => MinigamesPage()));
              }),
              _buildButton(context, 'Feedback', () {
                Navigator.push(
                    context, MaterialPageRoute(builder: (context) => FeedbackPage()));
              }),
            ],
          ),
        ),
      ),
    );
  }

  Widget _buildButton(BuildContext context, String text, VoidCallback onPressed) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 6.0),
      child: SizedBox(
        width: double.infinity,
        child: ElevatedButton(
          onPressed: onPressed,
          style: ElevatedButton.styleFrom(
            padding: EdgeInsets.symmetric(vertical: 16),
            textStyle: TextStyle(fontSize: 16),
          ),
          child: Text(text),
        ),
      ),
    );
  }
}
```

4.2.9 *Group* Pendalaman Alkitab

Pada segmen *group* pendalaman Alkitab, *user* dapat membuat grup, mengundang *user* lain dan berinteraksi dengan *user* lain melalui grup *chat*.

Segmen Program 4.11 *Code flutter* fitur *group* pendalaman Alkitab

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:provider/provider.dart';
import 'function.dart';
import 'user.dart';

class GroupListPage extends StatefulWidget {
  final List<Map<String, String>> groups;

  GroupListPage({required this.groups});

  @override
  _GroupListPageState createState() => _GroupListPageState();
}

class _GroupListPageState extends State<GroupListPage> {
  late List<Map<String, String>> _groups;
  String username = "";


  @override
  void initState() async {
    super.initState();
    username = context.read<UserModel>().username;
    _groups = widget.groups;
  }

  void _addGroup(String name, String description, String username)
async {
    try {
      await addGroup(name, username, description);
      setState(() {
        _groups.add({'name': name, 'description': description});
      });
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Gagal membuat Group')),
      );
    }
  }

  void _updateMemberStatus(String namaGroup, String username) async {
    await updateMemberStatus(namaGroup, username, "true");
  }
  void _removeMember(String namaGroup, String username) async {
```

```
    await removeMember(namaGroup,username);
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Bible Chat Groups'),
      ),
      body: ListView.builder(
        itemCount: _groups.length,
        itemBuilder: (context, index) {
          return Card(
            margin:  EdgeInsets.symmetric(vertical:  8,  horizontal:
16),
            child: ListTile(
              title: Text(_groups[index]['name']!,
                  style: TextStyle(fontWeight: FontWeight.bold)),
              subtitle: Text(_groups[index]['description']!),
              trailing: _groups[index]['status'] != "false"
                  ? Icon(Icons.chat)
                  : Row(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        TextButton(
                          onPressed: () => {

_updateMemberStatus(_groups[index]['name']!,username),
                            _groups[index]['status'] = "true",
                            Navigator.of(context).pop(),

Navigator.push(context,MaterialPageRoute(builder:    (context)    =>
GroupListPage(groups: _groups)),)
                          },
                          child: Text('Accept'),
                        ),
                        TextButton(
                          onPressed: () => {

_removeMember(_groups[index]['name']!,username),
                            _groups.removeAt(index),
                            Navigator.of(context).pop(),

Navigator.push(context,MaterialPageRoute(builder:    (context)    =>
GroupListPage(groups: _groups)),)
                          },
                          child: Text('Decline'),
                        ),
                      ],
                    ),
              onTap: () async {
                Navigator.push(
                  context,
```

```
                  MaterialPageRoute(
                    builder: (context) => ChatRoomPage(
                      groupName: _groups[index]['name']!
                    ),
                  ),
                );
              },
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          showDialog(
            context: context,
            builder: (context) => AddGroupDialog(addGroup: _addGroup,
username: username,),
          );
        },
        child: Icon(Icons.add),
        tooltip: 'Add Group',
      ),
    );
  }
}

class AddGroupDialog extends StatefulWidget {
  final Function(String, String, String) addGroup;
  final String username;

  AddGroupDialog({required this.addGroup, required this.username});

  @override
  _AddGroupDialogState createState() => _AddGroupDialogState();
}

class _AddGroupDialogState extends State<AddGroupDialog> {
  final _formKey = GlobalKey<FormState>();
  String _groupName = '';
  String _groupDescription = '';

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Buat Group'),
      content: Form(
        key: _formKey,
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextFormField(
              decoration: InputDecoration(labelText:'Nama Grup'),
```

```
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Masukkan nama grup';
                }
                return null;
              },
              onSaved: (value) {
                _groupName = value!;
              },
            ),
            TextFormField(
              decoration:    InputDecoration(labelText:    'Deskripsi
Grup'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Masukkan deskripsi grup';
                }
                return null;
              },
              onSaved: (value) {
                _groupDescription = value!;
              },
            ),
          ],
        ),
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.of(context).pop();
          },
          child: Text('Cancel'),
        ),
        TextButton(
          onPressed: () {
            if (_formKey.currentState!.validate()) {
              _formKey.currentState!.save();
              widget.addGroup(_groupName,          _groupDescription,
widget.username);
              Navigator.of(context).pop();
            }
          },
          child: Text('Add'),
        ),
      ],
    );
  }
}

class ChatRoomPage extends StatefulWidget {
  final String groupName;
```

```
  ChatRoomPage({required this.groupName});

  @override
  _ChatRoomPageState createState() => _ChatRoomPageState();
}

class _ChatRoomPageState extends State<ChatRoomPage> {
  final TextEditingController _controller = TextEditingController();

  void _inviteMember() async {
    try {
      List<String>            allMembers            =            await
getAllMember(context.read<UserModel>().username, widget.groupName);
      Navigator.push(
        context,
        MaterialPageRoute(builder:            (context)            =>
InvitePage(allMembers: allMembers, groupName: widget.groupName,)),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Failed to fetch members: $e')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.groupName),
        actions: [
          IconButton(
            icon: Icon(Icons.person_add),
            onPressed: () {
              _inviteMember();
            },
          ),
        ],
      ),
      body: Column(
        children: [
          Expanded(
            child: StreamBuilder<QuerySnapshot>(
              stream: getChat(widget.groupName, "grup"),
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  List<Map<String, String>> messages = [];
                  for (var doc in snapshot.data!.docs) {
                    Map<String, String> message = {
                      'groupName': doc['namaGroup'] ?? '',
                      'username': doc['username'] ?? '',
                      'chat': doc['chat'] ?? '',
```

```
                    };
                    messages.add(message);
                  }
                  return ListView.builder(
                    itemCount: messages.length,
                    itemBuilder: (context, index) {
                      bool  isSentByMe  =  messages[index]['username']
== context.read<UserModel>().username;
                      return Container(
                        margin:   EdgeInsets.symmetric(vertical:   5,
horizontal: 10),
                        alignment: isSentByMe ? Alignment.centerRight
: Alignment.centerLeft,
                        child: Bubble(
                          message: messages[index]['chat'] ?? '',
                          username:   messages[index]['username']   ??
'',
                          isSentByMe: isSentByMe,
                        ),
                      );
                    },
                  );
                } else if (snapshot.hasError) {
                  return          Center(child:          Text('Error:
${snapshot.error}'));
                } else {
                  return Center(child: CircularProgressIndicator());
                }
              },
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Row(
              children: [
                Expanded(
                  child: TextField(
                    controller: _controller,
                    decoration: InputDecoration(
                      labelText: 'Enter your message',
                      border: OutlineInputBorder(),
                    ),
                  ),
                ),
                IconButton(
                  icon: Icon(Icons.send),
                  onPressed: _sendMessage,
                ),
              ],
            ),
          ),
        ],
```

```
      ),
    );
  }

  void _sendMessage() async {
    if (_controller.text.isNotEmpty) {
      String username = context.read<UserModel>().username;
      String messageText = _controller.text;
      try {
        await addGroupChat(widget.groupName, username, messageText);
        _controller.clear();
      } catch (e) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Failed to send message: $e')),
        );
      }
    }
  }
}

class Bubble extends StatelessWidget {
  final String message;
  final String username;
  final bool isSentByMe;

  Bubble({required this.message, required this.username, required
this.isSentByMe});

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.symmetric(vertical: 10, horizontal: 15),
      decoration: BoxDecoration(
        color: isSentByMe ? Colors.blue : Colors.grey[300],
        borderRadius: BorderRadius.circular(10),
      ),
      child: Column(
        crossAxisAlignment:
            isSentByMe       ?       CrossAxisAlignment.end       :
CrossAxisAlignment.start,
        children: [
          Text(
            username,
            style: TextStyle(
              fontWeight: FontWeight.bold,
              color: isSentByMe ? Colors.white : Colors.black,
            ),
          ),
          Text(
            message,
            style: TextStyle(
              color: isSentByMe ? Colors.white : Colors.black,
```

```
            ),
          ),
        ],
      ),
    );
  }
}

class InvitePage extends StatefulWidget {
  final List<String> allMembers;
  final String groupName;

  InvitePage({required this.allMembers, required this.groupName});

  @override
  _InvitePageState createState() => _InvitePageState();
}

class _InvitePageState extends State<InvitePage> {
  List<String> selectedMembers = [];
  List<String> filteredMembers = [];

  @override
  void initState() {
    super.initState();
    filteredMembers = widget.allMembers;
  }

  void _filterMembers(String query) {
    final filtered = widget.allMembers.where((member) {
      return member.toLowerCase().contains(query.toLowerCase());
    }).toList();

    setState(() {
      filteredMembers = filtered;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Invite'),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              onChanged: _filterMembers,
              decoration: InputDecoration(
                labelText: 'Search',
```

```
                border: OutlineInputBorder(),
              ),
            ),
          ),
          Expanded(
            child: ListView.builder(
              itemCount: filteredMembers.length,
              itemBuilder: (context, index) {
                return CheckboxListTile(
                  title: Text(filteredMembers[index]),
                  value:
selectedMembers.contains(filteredMembers[index]),
                  onChanged: (bool? value) {
                    setState(() {
                      if (value == true) {
                        selectedMembers.add(filteredMembers[index]);
                      } else {

selectedMembers.remove(filteredMembers[index]);
                      }
                    });
                  },
                );
              },
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: ElevatedButton(
              onPressed: () {
                _inviteMembers(selectedMembers, widget.groupName);
              },
              child: Text('Invite'),
              style: ElevatedButton.styleFrom(
                padding: EdgeInsets.symmetric(vertical: 16),
                minimumSize: Size(double.infinity, 0),
              ),
            ),
          ),
        ],
      ),
    );
  }

  void _inviteMembers(List<String> selectedMembers, String groupName)
async {
    try {
      for (String member in selectedMembers) {
        await addMember(groupName, member);
      }
      Navigator.pop(context);
    } catch (e) {
```

```
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Failed to invite members: $e')),
      );
    }
  }
}
```

4.2.10 Forum

Dalam segmen *forum*, *user* dapat membuat post dalam bentuk *text* dengan mengisi judul dan deskripsi. *User* juga dapat melihat *post user* lain serta memberi komentar

Segmen Program 4.12 *Code flutter* fitur *forum*

```
import 'package:flutter/material.dart';
import 'function.dart';
import 'user.dart';
import 'package:intl/intl.dart';
import 'package:provider/provider.dart';


class ForumPage extends StatefulWidget {
  @override
  _ForumPageState createState() => _ForumPageState();
}

class _ForumPageState extends State<ForumPage> {

  void _showAddPostDialog() {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AddPostDialog(
          onAddPost: (judulPost, isiPost) async {
            await          addPost(context.read<UserModel>().username,
judulPost, isiPost);
          },
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Forum'),
      ),
      body: StreamBuilder<List<Map<String, dynamic>>>(
        stream: getPosts(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) return CircularProgressIndicator();
```

```
          final posts = snapshot.data!;
          return ListView.builder(
            itemCount: posts.length,
            itemBuilder: (context, index) {
              final post = posts[index];
              return      PostCard(post:      post,      currentUser:
context.read<UserModel>().username);
            },
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _showAddPostDialog,
        child: Icon(Icons.add),
      ),
    );
  }
}

class AddPostDialog extends StatefulWidget {
  final void Function(String, String) onAddPost;

  AddPostDialog({required this.onAddPost});

  @override
  _AddPostDialogState createState() => _AddPostDialogState();
}

class _AddPostDialogState extends State<AddPostDialog> {
  final _judulPostController = TextEditingController();
  final _isiPostController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Add Post'),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          TextField(
            controller: _judulPostController,
            decoration: InputDecoration(labelText: 'Judul Post'),
          ),
          TextField(
            controller: _isiPostController,
            decoration: InputDecoration(labelText: 'Isi Post'),
          ),
        ],
      ),
      actions: <Widget>[
        TextButton(
          child: Text('Cancel'),
```

```
          onPressed: () {
            Navigator.of(context).pop();
          },
        ),
        TextButton(
          child: Text('Add'),
          onPressed: () {
            final judulPost = _judulPostController.text;
            final isiPost = _isiPostController.text;
            if (judulPost.isNotEmpty && isiPost.isNotEmpty) {
              widget.onAddPost(judulPost, isiPost);
              Navigator.of(context).pop();
            }
          },
        ),
      ],
    );
  }
}

class Post {
  final String title;
  final String content;
  final DateTime timestamp;
  int likes;
  List<Comment> comments;

  Post({
    required this.title,
    required this.content,
    required this.timestamp,
    this.likes = 0,
    this.comments = const [],
  });
}

class Comment {
  final String user;
  final String content;
  final DateTime timestamp;

  Comment({
    required this.user,
    required this.content,
    required this.timestamp,
  });
}

class PostCard extends StatelessWidget {
  final Map<String, dynamic> post;
  final String currentUser;
```

```
  PostCard({required this.post, required this.currentUser});

  @override
  Widget build(BuildContext context) {
    final     formattedDate     =     DateFormat('dd     MMMM     yyyy,
HH:mm').format(post['timestamp']);
    return Card(
      margin: EdgeInsets.all(10),
      child: Padding(
        padding: EdgeInsets.all(15),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Text(
              post['judulPost'],
              style:      TextStyle(fontSize:      18,      fontWeight:
FontWeight.bold),
            ),
            SizedBox(height: 10),
            Text(post['isiPost']),
            SizedBox(height: 10),
            Text(
              formattedDate,
              style: TextStyle(color: Colors.grey[600]),
            ),
            ButtonBar(
              alignment: MainAxisAlignment.start,
              children: <Widget>[
                TextButton.icon(
                  icon: Icon(Icons.comment),
                  label: Text('Comment'),
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => CommentsPage(
                          postID: post['uniqueID'],
                          currentUser: currentUser,
                        ),
                      ),
                    );
                  },
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

```
class CommentsPage extends StatelessWidget {
  final String postID;
  final String currentUser;

  CommentsPage({required this.postID, required this.currentUser});

  void _showAddCommentDialog(BuildContext context) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AddCommentDialog(
          onAddComment: (content) async {
            await addComment(postID, currentUser, content);
          },
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Comments'),
      ),
      body: StreamBuilder<List<Map<String, dynamic>>>(
        stream: getComments(postID),
        builder: (context, snapshot) {
          if (!snapshot.hasData) return CircularProgressIndicator();
          final comments = snapshot.data!;
          return ListView.builder(
            itemCount: comments.length,
            itemBuilder: (context, index) {
              final comment = comments[index];
              final formattedDate = DateFormat('dd MMMM yyyy,
HH:mm').format(comment['timestamp']);
              return ListTile(
                title: Text(comment['username']),
                subtitle: Text(comment['comment']),
                trailing: Text(formattedDate),
              );
            },
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => _showAddCommentDialog(context),
        child: Icon(Icons.add_comment),
      ),
    );
  }
}
```

```
class AddCommentDialog extends StatefulWidget {
  final void Function(String) onAddComment;

  AddCommentDialog({required this.onAddComment});

  @override
  _AddCommentDialogState createState() => _AddCommentDialogState();
}

class _AddCommentDialogState extends State<AddCommentDialog> {
  final _contentController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Add Comment'),
      content: TextField(
        controller: _contentController,
        decoration: InputDecoration(labelText: 'Comment'),
      ),
      actions: <Widget>[
        TextButton(
          child: Text('Cancel'),
          onPressed: () {
            Navigator.of(context).pop();
          },
        ),
        TextButton(
          child: Text('Add'),
          onPressed: () {
            final content = _contentController.text;
            if (content.isNotEmpty) {
              widget.onAddComment(content);
              Navigator.of(context).pop();
            }
          },
        ),
      ],
    );
  }
}
```

4.2.11 *Feedback*

Dalam segmen *feedback, user* dapat memberikan kritik maupun saran mengenai aplikasi.

Segmen Program 4.13 *Code flutter* fitur *feedback*

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
```

```
import 'function.dart';
import 'user.dart';
import 'landing_page.dart';

class FeedbackPage extends StatefulWidget {
  @override
  _FeedbackPageState createState() => _FeedbackPageState();
}

class _FeedbackPageState extends State<FeedbackPage> {
  final _formKey = GlobalKey<FormState>();
  final _kritik = TextEditingController();
  final _saran = TextEditingController();

  void _submitFeedback(String kritik, String saran) async {
    if (_formKey.currentState!.validate()) {
      await                addFeedback(kritik,                saran,
context.read<UserModel>().username);
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: Text('Feedback Terkirim'),
            content:  Text('Terima  kasih  atas  kritik  dan  saran
Anda.'),
            actions: <Widget>[
              TextButton(
                child: Text('Oke'),
                onPressed: () {
                  Navigator.of(context).pop();
                  Navigator.push(
                    context,
                    MaterialPageRoute(builder:      (context)      =>
LandingPage()),
                  );
                },
              ),
            ],
          );
        },
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Feedback Page'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
```

```
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Text(
                'Kritik',
                style:    TextStyle(fontSize:    20.0,    fontWeight:
FontWeight.bold),
              ),
              SizedBox(height: 10.0),
              TextFormField(
                controller: _kritik,
                decoration: InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Masukkan kritik Anda',
                ),
                maxLines: 5,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Kritik tidak boleh kosong';
                  }
                  return null;
                },
              ),
              SizedBox(height: 20.0),
              Text(
                'Saran',
                style:    TextStyle(fontSize:    20.0,    fontWeight:
FontWeight.bold),
              ),
              SizedBox(height: 10.0),
              TextFormField(
                controller: _saran,
                decoration: InputDecoration(
                  border: OutlineInputBorder(),
                  labelText: 'Masukkan saran Anda',
                ),
                maxLines: 5,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Saran tidak boleh kosong';
                  }
                  return null;
                },
              ),
              SizedBox(height: 20.0),
              Center(
                child: ElevatedButton(
                  onPressed: () {
                    _submitFeedback(
                      _kritik.text,
```

```
                    _saran.text,
                  );
                },
                child: Text('Submit'),
                style: ElevatedButton.styleFrom(
                  padding: EdgeInsets.symmetric(vertical: 16),
                  minimumSize: Size(double.infinity, 0),
                )
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

4.2.12 *Minigames*

Dalam segmen *minigames*, *user* dapat memilih dan memainkan 2 *minigames,* yaitu *Bible quiz* dan *Bible search word*.

Segmen Program 4.14 *Code flutter* fitur *minigames*

```
import 'package:flutter/material.dart';
import 'dart:async';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'dart:math';

class MinigamesPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Minigames'),
      ),
      body: Center(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Expanded(
                child: Center(
                  child: ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.white,
                      foregroundColor: Colors.blue,
                      shadowColor: Colors.grey,
                      elevation: 5,
                      shape: RoundedRectangleBorder(
```

```
                        borderRadius: BorderRadius.circular(10),
                      ),
                      minimumSize: Size(double.infinity, 150),
                    ),
                    onPressed: () {

Navigator.push(context,MaterialPageRoute(builder:     (context)     =>
BibleSearchWordPage()),);
                    },
                    child: Column(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Image.asset(
                          'assets/logoWord.png',
                          height: 100,
                        ),
                        SizedBox(height: 10),
                        Text(
                          'Bible Search Word',
                          style: TextStyle(fontSize: 20),
                        ),
                      ],
                    ),
                  ),
                ),
              ),
              SizedBox(height: 5),
              Expanded(
                child: Center(
                  child: ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.white,
                      foregroundColor: Colors.blue,
                      shadowColor: Colors.grey,
                      elevation: 5,
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      minimumSize: Size(double.infinity, 150),
                    ),
                    onPressed: () {

Navigator.push(context,MaterialPageRoute(builder:     (context)     =>
BibleQuizPage()),);
                    },
                    child: Column(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Image.asset(
                          'assets/logoQuiz.png',
                          height: 100,
                        ),
```

```
                        SizedBox(height: 10),
                        Text(
                          'Bible Quiz',
                          style: TextStyle(fontSize: 20),
                        ),
                      ],
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

4.2.13 *Chat with Bible Bot*

Dalam segmen *chat with Bible bot*, *user* dapat bertanya kepada *Bible bot* dan *Bible bot* akan menjawab pertanyaan *user*.

Segmen Program 4.15 *Code flutter* fitur *chat with bible bot*

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'user.dart';
import 'function.dart';

class Chatbot extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Bible Bot'),
      ),
      body: ChatbotMain(),
    );
  }
}

class ChatbotMain extends StatefulWidget {
  @override
  _ChatbotMainState createState() => _ChatbotMainState();
}

class _ChatbotMainState extends State<ChatbotMain> {
  final TextEditingController _controller = TextEditingController();
  final ScrollController _scrollController = ScrollController();
  String greeting = "";
```

```
  String errorMessage = "";
  bool _isLoading = false;

  Future<void> _sendMessage(String text) async {
    if (text.trim().isEmpty) return;
    final username = context.read<UserModel>().username;
    String texts = text;
    _controller.clear();
    await addChat(username, 'user', texts);
    setState(() {
      _isLoading = true;
    });
    final botResponse = await getBotResponse(texts);
    print(botResponse);
    await addChat(username, 'chatbot', botResponse);
    setState(() {
      _isLoading = false;
    });
    _scrollToBottom();
  }

  void _scrollToBottom() {
    WidgetsBinding.instance.addPostFrameCallback((_) {
      _scrollController.animateTo(
        _scrollController.position.maxScrollExtent,
        duration: Duration(milliseconds: 300),
        curve: Curves.easeOut,
      );
    });
  }

  Widget _buildMessage(Map<String, dynamic> message) {
    final isUser = message['role'] == 'user';
    final alignment = isUser ? CrossAxisAlignment.end :
CrossAxisAlignment.start;
    final color = isUser ? Colors.blue : Colors.grey;

    return Column(
      crossAxisAlignment: alignment,
      children: [
        Container(
          margin: EdgeInsets.symmetric(vertical: 5),
          padding: EdgeInsets.symmetric(vertical: 10, horizontal:
15),
          decoration: BoxDecoration(
            color: color,
            borderRadius: BorderRadius.circular(15),
          ),
          child: Text(
            message['chat'],
            style: TextStyle(color: Colors.white),
          ),
```

```
        ),
      ],
    );
  }

  @override
  Widget build(BuildContext context) {
    final username = context.read<UserModel>().username;

    return Column(
      children: [
        Expanded(
          child: StreamBuilder(
            stream: getChat(username, "personal"),
            builder: (context, AsyncSnapshot<QuerySnapshot> snapshot)
{
              if (!snapshot.hasData) {
                return Center(child: CircularProgressIndicator());
              }
              final messages = snapshot.data!.docs
                  .map((doc) => {
                        'chat': doc['chat'],
                        'role': doc['role'],
                        'timestamp': doc.id,
                      })
                  .toList();
              WidgetsBinding.instance.addPostFrameCallback((_)     =>
_scrollToBottom());
              return ListView.builder(
                controller: _scrollController,
                padding: EdgeInsets.all(10),
                itemCount: messages.length,
                itemBuilder: (context, index) {
                  return _buildMessage(messages[index]);
                },
              );
            },
          ),
        ),
        if (_isLoading)
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Row(
              children: [
                CircularProgressIndicator(),
                SizedBox(width: 10),
                Text('Bible bot sedang berpikir...'),
              ],
            ),
          ),
        Padding(
          padding: const EdgeInsets.all(8.0),
```

```
          child: Row(
            children: [
              Expanded(
                child: TextField(
                  controller: _controller,
                  decoration: InputDecoration(
                    hintText: 'Type a message...',
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(20),
                    ),
                  ),
                  onSubmitted: _sendMessage,
                ),
              ),
              SizedBox(width: 8),
              IconButton(
                icon:             Icon(Icons.send,             color:
Theme.of(context).primaryColor),
                onPressed: () => _sendMessage(_controller.text),
              ),
            ],
          ),
        ),
      ],
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    _scrollController.dispose();
    super.dispose();
  }
}
```

4.2.14 *Bible Quiz*

Dalam segmen *Bible quiz*, *user* dapat bermain *game* yang dimana *user* diminta untuk menjawab 10 pertanyaan tentang Alkitab dengan waktu 10 detik tiap pertanyaan

Segmen Program 4.16 *Code flutter* fitur *bible quiz*

```
class BibleQuizPage extends StatefulWidget {
  @override
  _BibleQuizPageState createState() => _BibleQuizPageState();
}

class _BibleQuizPageState extends State<BibleQuizPage> {
  void   showPopup(BuildContext   context,   String   text,   {Duration
duration = const Duration(seconds: 2)}) {
    showDialog(
      context: context,
```

```
      barrierDismissible: false,
      builder: (context) {
        Future.delayed(duration, () {
          Navigator.of(context).pop();
        });
        return Center(
          child: AlertDialog(
            title: Text('Result'),
            content: Text(text),
          ),
        );
      },
    );
  }

  List<Map<String, dynamic>> questions = [];
  int currentQuestionIndex = 0;
  int score = 0;
  Timer? _timer;
  int _timeRemaining = 10;

  @override
  void initState() {
    super.initState();
    getDataPertanyaan();
  }

  Future<void> getDataPertanyaan() async {
    try {
      final QuerySnapshot<Map<String, dynamic>> snapshot =
          await
FirebaseFirestore.instance.collection('databaseBibleQuiz').get();
      final     List<QueryDocumentSnapshot<Map<String,     dynamic>>>
documents =
          snapshot.docs;

      Set<int> randomIndices = {};
      while (randomIndices.length < 10) {
        randomIndices.add(Random().nextInt(documents.length));
      }

      List<Map<String,       dynamic>>       fetchedQuestions       =
randomIndices.map((index) {
        final docData = documents[index].data();
        return {
          'question': docData['pertanyaan'],
          'answers': docData['jawaban'].split('|'),
          'correctAnswer': docData['kunci']
        };
      }).toList();

      setState(() {
```

```
        questions = fetchedQuestions;
      });

      startTimer();
    } catch (error) {
      print("Error: $error");
    }
  }

  void startTimer() {
    _timer?.cancel();
    _timeRemaining = 10;
    _timer = Timer.periodic(Duration(seconds: 1), (timer) {
      setState(() {
        if (_timeRemaining > 0) {
          _timeRemaining--;
        } else {
          timer.cancel();
          nextQuestion();
        }
      });
    });
  }

  void nextQuestion() {
    setState(() {
      if (currentQuestionIndex < questions.length - 1) {
        currentQuestionIndex++;
        startTimer();
      } else {
        _timer?.cancel();
        Future.delayed(Duration(seconds: 2), () {
          showQuizEndDialog();
        });
      }
    });
  }

  void checkAnswer(String answer) {
    String                          kunciJawaban                          =
questions[currentQuestionIndex]['correctAnswer'] as String;
    if (answer == kunciJawaban) {
      score++;
      showPopup(context, "Jawaban Anda Benar");
    } else {
      showPopup(context, "Jawaban Anda Salah\nJawaban yang benar :
$kunciJawaban");
    }
    Future.delayed(Duration(seconds: 2), () {
      nextQuestion();
    });
  }
```

```
  void showQuizEndDialog() {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: Text('Quiz Finished!'),
        content: Text('Your score is $score.'),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
              getDataPertanyaan();
              setState(() {
                currentQuestionIndex = 0;
                score = 0;
              });
            },
            child: Text('Play Again'),
          ),
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
              Navigator.of(context).pop();
            },
            child: Text('Back to Minigames'),
          ),
        ],
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Bible Quiz'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: questions.isEmpty
            ? Center(child: CircularProgressIndicator())
            : Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Row(
                    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                    children: [
                      Text(
                        'Score: $score',
                        style:  TextStyle(fontSize:  20,  fontWeight:
FontWeight.bold),
```

```
                    ),
                    Text(
                      'Question: ${currentQuestionIndex + 1} of
${questions.length}',
                      style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
                    ),
                    Text(
                      'Time: $_timeRemaining',
                      style: TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
                    ),
                  ],
                ),
                SizedBox(height: 20),
                Expanded(
                  child: Center(
                    child: Text(
                      questions[currentQuestionIndex]['question']
as String,
                      style: TextStyle(fontSize: 24),
                      textAlign: TextAlign.center,
                    ),
                  ),
                ),
                Column(
                  children:
(questions[currentQuestionIndex]['answers']                          as
List<String>).map((answer) {
                    return Container(
                      width: double.infinity,
                      margin: const EdgeInsets.symmetric(vertical:
5),
                      child: ElevatedButton(
                        onPressed: () => checkAnswer(answer),
                        style: ElevatedButton.styleFrom(
                          padding:   EdgeInsets.symmetric(vertical:
15),
                          textStyle: TextStyle(fontSize: 18),
                        ),
                        child: Text(answer),
                      ),
                    );
                  }).toList(),
                ),
              ],
            ),
      ),
    );
  }

  @override
```

```
  void dispose() {
    _timer?.cancel();
    super.dispose();
  }
}
```

4.2.15 *Bible Search Word*

Dalam segmen *Bible search word*, *user* dapat bermain *game* yang dimana *user* diminta untuk mencari 5 kata tentang Alkitab dan *user* dapat melihat *clue* dengan menekan tombol lampu.

Segmen Program 4.17 *Code flutter* fitur *bible search word*

```
class BibleSearchWordPage extends StatefulWidget {
  @override
  _BibleSearchWordPageState              createState()              =>
_BibleSearchWordPageState();
}

class _BibleSearchWordPageState extends State<BibleSearchWordPage> {
  int _score = 0;
  String _selectedText = '';
  List<int> _selectedIndices = [];
  List<int> _lockedIndices = [];
  int _timeLeft = 120;
  Timer? _timer;
  String _currentClue = '';
  List<String> _keyAnswers = [];
  List<String> _characters = [];

  @override
  void initState() {
    super.initState();
    getDataSearchWord();
  }

  Future<void> getDataSearchWord() async {
    try {
      final QuerySnapshot<Map<String, dynamic>> snapshot =
          await
FirebaseFirestore.instance.collection('databaseSearchWord').get();
      final     List<QueryDocumentSnapshot<Map<String,     dynamic>>>
documents =
          snapshot.docs;
      if (documents.isNotEmpty) {
        final randomDocIndex = Random().nextInt(documents.length);
        final randomDoc = documents[randomDocIndex];
        List<String>                  mapCharacters                =
randomDoc.data()['map'].split('');
        _characters.addAll(mapCharacters);
```

```
        List<String>                    answers                    =
randomDoc.data()['jawaban'].split('|');
        _keyAnswers.addAll(answers);
        _currentClue = randomDoc.data()['clue'];
        startTimer();
      } else {
        print('No data found');
      }
    } catch (error) {
      print("Error: $error");
    }
  }
  void startTimer() {
    _timer = Timer.periodic(Duration(seconds: 1), (timer) {
      if (_timeLeft > 0) {
        setState(() {
          _timeLeft--;
        });
      } else {
        timer.cancel();
        showEndDialog(false);
      }
    });
  }

  void onButtonPress(int index) {
    if (!_lockedIndices.contains(index)) {
      setState(() {
        if (_selectedIndices.isEmpty ||
            (_selectedIndices.last ~/ 8 == index ~/ 8 &&
                (_selectedIndices.last - index).abs() == 1) ||
            (_selectedIndices.last % 8 == index % 8 &&
                (_selectedIndices.last - index).abs() == 8)) {
          _selectedText += _characters[index];
          _selectedIndices.add(index);
        }

        if (_keyAnswers.contains(_selectedText)) {
          _score += 1;
          _lockedIndices.addAll(_selectedIndices);
          _selectedIndices.clear();
          _selectedText = '';

          if (_score >= 5) {
            _timer?.cancel();
            showEndDialog(true);
          }
        }
      });
    }
  }
```

```
  void showEndDialog(bool isWin) {
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (BuildContext context) {
        return AlertDialog(
          title:  Text(isWin  ?  'Selamat,  kamu  berhasil'  :  'Waktu
habis'),
          content: Text(isWin
              ? 'Selamat, kamu telah menyelesaikan permainan ini'
              : 'Nice Try, coba lagi nanti'),
          actions: [
            TextButton(
              child: Text('Play Again'),
              onPressed: () {
                Navigator.of(context).pop();
                resetGame();
              },
            ),
            TextButton(
              child: Text('Back to Minigames'),
              onPressed: () {
                Navigator.of(context).pop();
                Navigator.of(context).pop();
              },
            ),
          ],
        );
      },
    );
  }

  void resetGame() {
    getDataSearchWord();
    setState(() {
      _score = 0;
      _selectedText = '';
      _selectedIndices.clear();
      _lockedIndices.clear();
      _timeLeft = 120;
      _characters.clear();
      _keyAnswers.clear();
    });
  }

  void showClueDialog() {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text('Clue'),
          content: Text(_currentClue),
```

```
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text('Close'),
            ),
          ],
        );
      },
    );
  }

  @override
  void dispose() {
    _timer?.cancel();
    super.dispose();
  }

  void clearText(bool resetGame) {
    setState(() {
      _selectedText = '';
      _selectedIndices.clear();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Bible Search Word'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Text(
                  'Score: $_score',
                  style:    TextStyle(fontSize:    20,    fontWeight:
FontWeight.bold),
                ),
                IconButton(
                  icon: Icon(Icons.lightbulb_outline),
                  onPressed: () {
                    showClueDialog();
                  },
                ),
                Text(
```

```
                  'Time: $_timeLeft',
                  style:    TextStyle(fontSize:    20,    fontWeight:
FontWeight.bold),
                ),
              ],
            ),
            SizedBox(height: 20),
            Container(
              padding: EdgeInsets.all(10.0),
              decoration: BoxDecoration(
                border: Border.all(color: Colors.black),
                borderRadius: BorderRadius.circular(5.0),
              ),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    _selectedText,
                    style: TextStyle(fontSize: 20),
                  ),
                  _selectedText.isNotEmpty
                      ? IconButton(
                          icon: Icon(Icons.clear),
                          onPressed: () => clearText(false),
                        )
                      : SizedBox(),
                ],
              ),
            ),
            SizedBox(height: 20),
            Expanded(
              child: GridView.builder(
                gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(
                  crossAxisCount: 8,
                  mainAxisSpacing: 0,
                  crossAxisSpacing: 0,
                  childAspectRatio: 1,
                ),
                itemCount: _characters.length,
                itemBuilder: (BuildContext context, int index) {
                  return GestureDetector(
                    onTap: () => onButtonPress(index),
                    child: Container(
                      decoration: BoxDecoration(
                        border: Border.all(color: Colors.black),
                        color: _lockedIndices.contains(index)
                            ? Colors.red
                            : _selectedIndices.contains(index)
                                ? Colors.green
                                : Colors.blue,
                      ),
```

```
                      alignment: Alignment.center,
                      child: Text(
                        _characters[index],
                        style:    TextStyle(fontSize:    20,    color:
Colors.white),
                      ),
                    ),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```