

```
└─ ./
   .gitattributes
   .gitignore
   DejaVuSans.ttf
   final_output.pdf
   project_to_pdf.py
   └─ .git/
      .COMMIT_EDITMSG.swp
      .MERGE_MSG.swp
      COMMIT_EDITMSG
      config
      description
      FETCH_HEAD
      HEAD
      index
      ORIG_HEAD
      └─ hooks/
         applypatch-msg.sample
         commit-msg.sample
         fsmonitor-watchman.sample
         post-update.sample
         pre-applypatch.sample
         pre-commit.sample
         pre-merge-commit.sample
         pre-push.sample
         pre-rebase.sample
         pre-receive.sample
         prepare-commit-msg.sample
         push-to-checkout.sample
         sendemail-validate.sample
         update.sample
      └─ info/
         exclude
      └─ logs/
         HEAD
         └─ refs/
            └─ heads/
               ai-vocabulary-generation
               login-register-feature
               main
               refactoring
               vocabulary
               watch_videos
            └─ remotes/
               └─ origin/
                  ai-vocabulary-generation
                  HEAD
                  login-register-feature
                  main
                  refactoring
                  vocabulary
                  watch_videos
      └─ objects/
```

— 00/
6517533656eda680c59f70d849b0342c2610e9
65b5eb3688479afe5e7cdc743f4389d60d3b77
8c2718817cb0b4923aa615e7efd56180cab6f9
919b35999e9e38324f5bfdaab3801f999be0a7
— 01/
7211ea4beddd40426a65f0f8287c87ead45eba
— 02/
09012906bd60d5cfd54e7f78d91204d70c5881
2bd3e3ba8789b175c86af33eefdf8f1fa57e1
422afb5e4a3b44b4a34ccb43c118b3563a6dea
— 03/
5496e63b56af046bd1286ed29c5ad981808e1d
— 05/
0e0cc407b0a2bed50f089188e78216d6015516
— 07/
80dcb3bbb5ee2ab087b8dbedbbede989869c1b
— 0b/
13cabe494601d35b9fb2d93a99c5fef0916827
30e10c4c042013ea942fd0d757042a52514f8c
— 0c/
26f970de0f3f09586b8800e973879078b5c001
36de6d19883e217e673c077d6c4b51f842445b
3ec5f18d760d7cd33b8e24962c1392d6ff124d
c68ad796a26df76448bee297d9b83e6cb1c13c
f7ce562d9e68f96551714dc2f55f63edc5aac6
— 0d/
5ca5da88199118bf964bb0d8fea0168d884d27
7993438f91b1577c483b53a7888258c805d9fa
— 0e/
207aab6d07ac09220f4d3c1074e475534b7c4f
— 0f/
df60bb4a32b7bf4422d2cde87860097e95c10f
— 11/
1ff6bd80fcb6cbd1e74b5fe5c90682c844b667
— 12/
36808fcc97581f59d9b784f97daf5389f68e05
4a2761a315e96cdea296a3840d895fe3398b03
9a187fd5b6bfe429628cd5b3abdbecf09cd1ec
— 13/
5d48e11591b43dd9418c047f328bdf419f8adb
— 14/
8d9500c9cda6308d39953f90fc67282d29ec4e
— 15/
2b8cdd9a34961d67615318d5b259cddcf674af
— 16/
2bfc2af942733303b86efacd2cbbee73eeaac9
— 17/
7854d8fb070915778ae3eada4fbca40af9d984
— 19/
04197267a5f000b457fd42929780fc07bf26a5
— 1b/
b55403fd60a518e3fa42bdcb836ecb82226750

— 1c/
28ebd2a56bc08f5d24e80b6ccafe4b0410b1f2
4382a0c2cf21edc14e94a90ddfb25442e16cdc
— 1d/
fa1357c137674bb62d137a13a50dff1ae780a
fb691d9b17a2e3765f1f5dc4bd586e46e6480e
— 1e/
b7aafb25731c127a3558267515efd496b698a8
— 1f/
3b1dcc453aba4b1bb6b9d6a0b5611916426468
b2c535203cd2872724912f740db3e60df188df
— 20/
491ed5bb61347da1ce961a9a771c4cf97542bd
— 22/
e2c6535f084532c36798012ebccfb7396d04f
— 23/
d0a34f1ea8d7133b10f2c1a11eaaebf4d21a3b
— 24/
388c0694b71d86c03226f43f9c16ed087a214c
9f9de8bcaebc28d5ba4584c33ef3d1a32205c2
— 26/
9ecd616f3c52b067e691114397f6a47a1aee4d
b0e491c0c0fb7fc8bd05a7efb69899da4a1930
— 28/
2646dcb22cef2f623b14e741f1047bcab939b0
dafb589910e2eb3c967dfcc2939d24ee1ebad4
— 2c/
91ec2ff8eb8bcee28bf8794c4341e0909dad9c
— 2f/
518741048fc52c609aa38319364e9edbb95f83
9f162f9224af4db9a881ce2ce1629e8e689657
— 31/
440a23beafbc3ecfa5e906dc41a63fd95ee31b
— 32/
8f7aee5436a382f62d92920b9de43cf320b055
ecf4290244fa7e06e69722d62ce072e28677a2
— 33/
01408c50e49807fabd69ef0cecae3e4ab5d2b0
16baa58dfed257815877e5ef8b48e0b18bf
— 37/
69c6b16becf4525b8277d976a5562cc64078eb
— 39/
126108ce721686d2ac8badf4137787d69f3c27
2f51940efa0662191f84155ff6eab74f014143
8511bc9cab44e252179209486ce68927658c3b
— 3a/
1b8da6b4c7ef90fcf9cf63c128d01c39d4b750
— 3c/
9b36c765ac387a8871a008f12ad91a43d2d88d
b94bc5bca6344e45a1cc63f2e1cfbb4a6fe0ca
dc24a45aaf813c46720f2282c40f98dd685c43
— 3e/
53aa29d8790b8bbf90d0abcd611f9e4a91a122

6187a8fc255d0939ab3e20963d14dc531ed4f2
— 3f/
8bfe99fd08f1e41f8b8a21b17d6b0c7fe3d777
— 41/
70cf6f9ef167d036071b81ffec588c550ffd8c
e93856985cc26ddcb71832e6bcb7e768ed78c
— 42/
cca7de0bb3bec81351e5611e30dc0e5442a4e0
— 46/
2afead77f485fd5ac57c173ef9437132456083
39a08f179604d605afcaeeae4457569acffc9b
— 4b/
e6e921a0ad8a2804fd2e3d0b27f591e384b53f
— 4d/
3b76edaf07feb8e0138e399dd78a88b2f65321
79767246372230f00e33b338d8aa93bf8fa7cd
c6cc75dae6f9fd99d971d6787b6ffece7b5dd2
daa9873f23907a3092c2246d79bdbb419cb065
ed3b7793a89a18582882849485d8142ec0a62a
— 4e/
16863c17a5685167067646195436938419c9be
5e57bef8db7c790cc57aa8b635fbd3afa6cd61
— 4f/
9a94876c6a464dda04ea16594e8d2e70c9a8a5
9bbdadff091c8e5240e8103eca70c4076d2b0b
— 50/
1067f252bd40268548a5255a9f9578e4e19a40
eb385a77d22bc38ce278b7b03b93ac012f0b1f
— 51/
5deab7629022bdb29df3e8e51d4fa08297bbb4
69aca394a9c1e06f309758f82ca4db190d97ff
— 52/
764f6785dfe61ef94b8b2728c0fe194bd31c74
— 53/
15950e34ed30f5c2258a3a713ec6b08569059b
— 54/
da60e427e67ccc97de404b4215f95b57025fc8
ef2084e7aca6233103e2d2fbf0ec45cae9f54f
— 55/
0f3c8de429e081c3cbd03279e287be7350ad7e
— 56/
9a403f10304ed93fbbf8257ee9e214e41d25b2
— 57/
b4ee1b0cb7c2579ad6071539e094f35da50fef
— 58/
c38ffd6a49b1284b979457ea76c53dae83c280
— 59/
60cb1e1b598ce11b34709583932e103d4c89fc
— 5b/
14bdae48f2638c9d12fb7597593f4363c0a0fa
291bf3f3f07780f889c7b08b90802eca31243d
690432f54f50b57b36b1267d44e617056774d3
a492b5aa6a9f81ded0c07288745e75aedabce0

ec82c53d08119b75363f436ac290bade5747ce
— 5e/
0e0fc8da6323e9370892e3945037ca1bda9cc0
cd356b90c109dceae7368e3b822155c4b3b5cd
— 60/
532798bd909f128de67c7885862c82e7910954
— 63/
6b2db6be3ee4463ef375aac6efedeee3640152
— 64/
60a5ae736675b80f63419d518f83fe6ef374aa
64e285ee2f2fc61f5c2725166d2a8e572de24b
92637607a9920e51f36ef2ce18883e4e21e721
bd95221a20e8b52452563dd199e7eaf36e68c7
— 65/
ca996e3f1599632f76d26074ac476ce05b8026
— 66/
ecb3c03a75b7f634f237a71e03b4357a2d36cb
— 67/
f5f63718c5769413f079ae9a43cee0b81a3753
— 68/
8f8542cd75d8b043afecf52ae2e92234b0ccad
— 69/
bd7a2bff2f0f5e04ec2d08ab98f2a0053a661e
bdca3633264e442360459e2449df3626e418a0
— 6b/
22ade825fd85a0f1a82f4341da5b54f5044968
— 6d/
fb8d9eb3dea17a8d9bae51867e7621a3bb60fe
— 6e/
1cd8d55a0f60f8858c4a667bee0ca9b7aa48a7
— 6f/
9509c88bed7080d496fc5e1d87a9315e30549d
b40c70afc06dd54ca3e556a7a518e42659286c
— 72/
1cf9ff01d8da234a36006691377a3165bfe7ce
2b8bf6b83de3920fc76f6c99b5d7e069b9428d
— 74/
2f90c5ee26340d333a79ed059447a9bfb9a407
3d61e01daa838673e0283cb01a0799aeeee1d3f
d9690b6d4d6a1c81fb212e88a319922345f733
— 76/
045c9331dcf0c817619bbbc908074c09626900
— 77/
9d49f6b2a848dc8e2265860ec75cf2c050359d
ea6320dd7fd57823e368199226f7a4a8247cd7
— 79/
3c25426714ebd13e56ad01525e329624485b4f
— 7b/
1249f9cc819b50c1348ccf7ba9e0d438653d3e
d7f826250e7064d5793ffa36d972c12f28c5ef
— 7f/
12a02584f4ba838241bddb9da62cdda4cc4a85
— 81/

038b052f1506d55be3ee420cbaf1adee9ea1b4

— 84/

2798fba922b7ad3815a4fd44b798661f438cae
7cb0adecd86a28b200600b0c10cab904892fc4
cefba8cd023cac1784c835bd62fa3dc4f5351e

— 85/

015afcc5b18fb8e78d44863501206e9fd9477f
0eddc73e06fa3f619f36f2eb541416e91b0acc

— 86/

7d5727a4d8b61320a69978fce399a72d2bd212
84db6cdadb5a8e390c4a47438622beb31e6402
8a637b051da6c8b101f800ab60bb68ee5ea2db

— 87/

132b042e96065ee35426f3f2085e8497269025

— 88/

c74c10851a8063ab69df82a5969f427fadb21d

— 8a/

c81a3c49d062a279d36018134519aed939151c

— 8b/

2820b754ab0100db9f9f1d24a585dcf82bea27
28301d6e8502a4e6a4ba6e745c97fb89269d7e
5e9f7e17f4529658f47034431897811b2193fe

— 8c/

89f4a0309a312f33371be10898ae413c1dde41

— 8d/

ddf3669982fa595af6bea63542894bb8013fe4

— 8f/

9f582ab2ecce1faa610426321a2a46dd6d3bb0

— 92/

0d21d196501b0895d400f5f3ae94ced8e3b715
393f41324db717e50c9ea35e349d0969170ae6
e931572ab1c02234f243a790a2d693712f619f

— 95/

4e8ef3c7722dca22768af4eb181d3b91333bec
68d7e82c8b5dd5578fa68d0798110f88f48f46

— 96/

8e433211f5c403512b39e7af9ff00f0d03b7b2

— 97/

3a0314015ec1b46235dee7509e518a24394b9a
9687755aa5345a63b4acf0b34fe8ded34e5bdf

— 98/

f31bcdb7261d2e4d0587564925bc3516c9a6f9

— 99/

66c1bf170c2d3bb63dd243909e9a85be6df22c

— 9c/

a7879dc3335d4d3f7deaf697b2d652121bac5c

— 9d/

0d4adff075868fce4aca071d72417686b6605b

— 9e/

0267e4e8c2f2d4d5ec6f3c809cc39da20751d2
ee275518aa5b6ea282f0ae616c2480f4da2d20

— 9f/

329eb66ddb57a8feaf0f61141450fe79bb85ef

— a0/
d7407156d2757d82c5c700b8ce8e41a4c13100

— a2/
68043ca587b0c4f83f7076e45a3503a90a6996
7f98f6b7cfae2185592a71c0b513934113fcf7

— a3/
1a4336fd1d5d7ba24c771df37047fa967782a0
ded5cbb298242b5f25470389c877a5a426db8

— a5/
c62b82db228b204a20a290fa5d487b96cf557f

— a6/
7df1f1cc80b6d9c9b3854236986d094fa3657f

— a8/
cf8d0ffc3caf9703078b712fbd020a2aaa474d

— a9/
3679ad2f71ae8531aaaae69614c83adf67ab3a

— aa/
493375f0ae3cec63051c7de5fe2eb3137c3d6d

— ab/
e0d69cd4604bd09c5facf1a012f2be062bac33
f34c0a615fa23925f1fe915979b36ff6913faa

— ac/
0cbf02a2a86f89c8bc460f9ed3171c526fc26b

— ad/
827407ca9956f35018ee425c8eb660bcf9e5d1

— af/
54e267b39268ed512a8a72a5b3fc8dc0db0c31

— b3/
656876b8affad87ef791e20ff0466ac292ed6a
d786489ece5f5deddfefec23740bfac5207af4

— b6/
1b44162a4db1c9c79e5b716d7fd4fc26f506c5
685de8ed28418237892ba15d36249d5636f86c
d487ba5c4082e600f41d49e85dceb8fa1d08cf

— b8/
2d87d49f9c2945e5292c14e261d60a401e8805

— b9/
56e1b55f462147b4142f855ecf60fda2654468

— bd/
98e9e023e8278df04314f84a3250184ef421a6

— bf/
053275f266be86b4f05d7283fdda9d3f032bcb

— c0/
9c9c0036c9b3daeb218f6af919783035f63668

— c1/
3e1ba88ee065299d353f1ae101c801d4a0cfd6

— c2/
2f1993c00680e5c3b217233ea93a38d86ea610
9c09f88c09ae1977b4281d646e1cd99eb540d5

— c4/
b02c595464071d520defd1f02e55cc22b77839

— c8/
1df22ef0ac8296f7460ab6103489e3831c1d03

891c2156f22a539b591a62576f2fcf6489d122
— c9/
119437c302d5ea6460ae9aabb18406abe92cf
— ca/
d80d4d8614cef96149bd5b9ed6f45f5ec924b4
dec6ae73df79a293d2dc21eb36a978235e9dfa
— cc/
31f7126189c3bbc7a6e5b50567ea9a2e4af042
— cd/
f86769ee079727ea4cc5a2bc2784f26d3f8908
— cf/
2fa27a0a38dca3a2527745d2d364b339c75aa3
385bd0c9bb61cbcf57798f68e690af59f41d26
— d0/
92f6c4c2a4684997596e0bb93045f50517f872
db4da2a5ba06b7b6c138f464a84fe6d2e3d821
fccfd73aa22d04694b38e887c2b6063c45bbaf
— d4/
18d680f683b2d07597abc5c10827769e73c7f5
352c74490b4955402c3cf3fdd9b463f93de046
76f7b96f5672ef8aab62793773e5caab2bcf46
— d5/
69fbdb5c2a2b55445ca9029b340adaf4043772
c017fd905c05398926d7e246b9063f9caac014
f59842ca9c27376709c4b881d52e45a64232bd
— d8/
184d028a8a50ee94636b82cdd5f14ae3623c12
776039e6e682f147b2a3db5dc203a13fcb7e3c
— d9/
f22f08f494e6c7850ac3180b8830f28e13e2d9
— db/
4f6cd25048bfb2cead2049c4efc042d40fe5fc
9bf1250682f63d7d3a3b6aca7509b01c5d96a4
— de/
b808071af217a8767e4d68d3bbf4db3d83d9dd
— df/
e0770424b2a19faf507a501ebfc23be8f54e7b
— e1/
2086927223f320493b805f03bc697253098143
— e2/
339dbe25ebb72242b8cc74ba43b76ff88fb0fa
— e3/
356862bfd2c8e8b8a6c40557d32b20ef339f61
— e4/
0fc2cd6418b97608d8c2828fc7a2b25518a622
3c5ac2c68d1f3bf422fbf2750e22d28a85b155
ac2e5ab2a5933db12806c81b8becb23fee0ee
— e5/
25e6d7839a48722c6e49a0d536a23095dae4c3
9a4c5f4239675ff85d24af359a933bd1e13927
babd58a1b8bb7f77f761f3f08da6465a3c1059
f7eecce43be41ff0703ed99e1553029b849f14
— e6/

- 9de29bb2d1d6434b8b29ae775ad8c2e48c5391
 - e7/
 - a5a13f5bf89070e7b6e1cbe985da50e7763dca
 - f08517e9939eb4981f437cf8548391625b0d26
 - ef/
 - 417601ddaf9e159d8540a7520b8512d777701e
 - 7d20c9ab4f9d933121705380f0af40cefa0554
 - f1/
 - 29bb3b1264fc8331504ad0e67609e88e982e11
 - f2/
 - 2c9cc73c7e5d34640e1058f8e09b8e315fb059
 - f5/
 - 1d5ff2eb7ecc32cfc3bfc4b9501de4c7f5fa77
 - fd46fe294445a78897767de35dbeddce3db0cb
 - f6/
 - 17be66486fd49c78313b1a5431181a8704fc58
 - ba88f6595730634e54411278dbdeca9a80127a
 - f7/
 - 0ecbcaec88b3e2aa5c48abf2852463864c42f3
 - 52e475dbda416e770d032c6a750657b2230deb
 - f8/
 - 10e00737bc3df8855bbdca8b362f38bd870657
 - f9/
 - 5b2f56a6ae96748259fdb91a4272bfb803c944
 - db032d3ac050aa543e3aa68f5d303cbcbcb2443
 - fa/
 - 73fc3d9f539f0233f4adb62e037af956175e15
 - fb/
 - b96afa731457bfdafdb646bdd1a9821ec36e40
 - eb7bc544177c343f80098ad2383c8d097705ad
 - ffe97b950c60d9025a9c59678954d9e6e4c582
 - fc/
 - 51eb38017a8b69b1c5af2a09b748fa5e1ffeed
 - fe/
 - 397c397cfe428a75b711902844caa2564196f0
 - 6a62b7677e12ff77b48bea916a2dbd9905f659
 - ff/
 - cb0b487a293073b1a6cd7d08bd517bb3ee3add
 - info/
 - pack/
 - refs/
 - heads/
 - ai-vocabulary-generation
 - login-register-feature
 - main
 - refactoring
 - vocabulary
 - watch_videos
 - remotes/
 - origin/
 - ai-vocabulary-generation
 - HEAD
 - login-register-feature

- | | | | main
- | | | | refactoring
- | | | | vocabulary
- | | | | watch_videos
- | | | — tags/
- | — config/
- | | config.php
- | | config.py
- | | keys.json
- | | __init__.py
- | | | — __pycache__/
- | | | | config.cpython-311.pyc
- | | | | __init__.cpython-311.pyc
- | — controller/
- | | AuthController.php
- | | LogoutController.php
- | | SaveProfileController.php
- | | SaveVocabController.php
- | — Diagrams/
- | | | — Activity Diagram/
- | | | | Login_Register Activity Diagram v0.0.png
- | | | — Structure/
- | | | | LingoLoop_Project_Structure.pdf
- | | | — Use Case/
- | | | | Use Case v0.0.pdf
- | — models/
- | | ai_title_generation.py
- | | ai_vocabulary_generation.py
- | | Database.php
- | | Database.py
- | | SaveProfile.php
- | | SaveVocab.php
- | | SessionManager.php
- | | User.php
- | | youtube.py
- | | | — __pycache__/
- | | | | ai_title_generation.cpython-311.pyc
- | | | | Database.cpython-311.pyc
- | — view/
- | | dashboard.php
- | | index.php
- | | login.php
- | | register.php
- | | select_video.php
- | | select_vocab.php
- | | setup_profile.php
- | | show_transcript.php
- | | watch_video.php
- | | welcome.php

project_to_pdf.py

```
import os
from fpdf import FPDF
from PyPDF2 import PdfMerger

# Font file must be in the same directory as this script
FONT_PATH = "DejaVuSans.ttf"
ROOT_DIR = "." # Root directory to scan
FINAL_PDF = "final_output.pdf"

# Generate a visual folder structure as a string
def generate_structure_text(root_dir):
    structure = ""
    for dirpath, dirnames, filenames in os.walk(root_dir):
        level = dirpath.replace(root_dir, "").count(os.sep)
        indent = " | " * level + " |—"
        structure += f"{indent}{os.path.basename(dirpath)}\n"

        subindent = " | " * (level + 1)
        for f in filenames:
            structure += f"{subindent}{f}\n"
    return structure

# Create a PDF from the folder structure string
def write_structure_to_pdf(text, output_file="structure.pdf"):
    pdf = FPDF()
    pdf.add_page()
    pdf.add_font("DejaVu", "", FONT_PATH)
    pdf.set_font("DejaVu", size=10)
    pdf.multi_cell(0, 5, text)
    pdf.output(output_file)
    print(f"Folder structure saved as: {output_file}")

# Convert individual code file into PDF (with filename as title)
def convert_file_to_pdf(file_path, output_path, font_file=FONT_PATH):
    try:
        with open(file_path, 'r', encoding="utf-8", errors="ignore") as f:
            content = f.read()
    except Exception as e:
        print(f"Could not read file {file_path}: {e}")
        return

    pdf = FPDF()
    pdf.add_page()
    pdf.add_font("DejaVu", "", font_file)

    # File name as PDF title
    pdf.set_font("DejaVu", size=14)
    pdf.cell(0, 10, os.path.basename(file_path), ln=True)

    # File content
    pdf.set_font("DejaVu", size=8)
```

```

pdf.multi_cell(0, 5, content)
pdf.output(output_path)
print(f"PDF created: {output_path}")

# Get all .py, .php, .html files recursively
def get_all_code_files(root, extensions=(".py", ".php", ".html")):
    collected = []
    for dirpath, _, filenames in os.walk(root):
        for f in filenames:
            if f.endswith(extensions):
                collected.append(os.path.join(dirpath, f))
    return collected

def main():
    # Step 1: Generate folder structure
    structure_text = generate_structure_text(ROOT_DIR)
    structure_pdf = "structure.pdf"
    write_structure_to_pdf(structure_text, structure_pdf)

    # Step 2: Convert all code files to individual PDFs
    code_files = get_all_code_files(ROOT_DIR)
    generated_pdfs = [structure_pdf]

    for file in code_files:
        filename = os.path.basename(file)
        output_pdf = filename + ".pdf"
        convert_file_to_pdf(file, output_pdf)
        generated_pdfs.append(output_pdf)

    # Step 3: Merge all PDFs into a single final document
    merger = PdfMerger()
    for pdf in generated_pdfs:
        if os.path.exists(pdf):
            merger.append(pdf)

    merger.write(FINAL_PDF)
    merger.close()
    print(f"Final PDF generated as: {FINAL_PDF}")

    # Step 4: Delete all temporary individual PDFs
    for pdf in generated_pdfs:
        if os.path.exists(pdf) and pdf != FINAL_PDF:
            os.remove(pdf)
            print(f"Deleted: {pdf}")

if __name__ == "__main__":
    main()

```

config.php

```
<?php
```

```
define('DB_HOST', 'localhost');  
define('DB_USER', 'root');  
define('DB_PASS', '');  
define('DB_NAME', 'lingoloop');
```

config.py

```
DB_CONFIG = {  
    'host': 'localhost',  
    'user': 'root',  
    'password': '',  
    'database': 'lingoloop'  
}
```

`__init__.py`

AuthController.php

```
<?php

require_once __DIR__ . '/../models/User.php';
require_once __DIR__ . '/../models/SessionManager.php';
require_once __DIR__ . '/../models/Database.php';

class AuthController {
    private User $userModel;

    public function __construct() {
        $db = Database::getInstance();
        $this->userModel = new User($db);
    }

    public function login(string $username, string $password): ?string {
        $user = $this->userModel->findByUsername($username);

        if (!$user) {
            return "User not found.";
        }

        if (password_verify($password, $user['password'])) {
            SessionManager::start($user['id'], $user['username']);

            // Provjeri da li korisnik već ima profil
            if (!$this->userModel->hasProfile($user['id'])) {
                header("Location: /lingoloop/view/setup_profile.php");

                exit();
            }

            $scriptPath = BASE_PATH . "/models/ai_vocabulary_generation.py";
            $command = "python \"$scriptPath\" { $user['id'] } 2>&1";

            $output = shell_exec($command);
            echo "<pre>";
            echo "PYTHON RAW OUTPUT:\n";
            echo htmlspecialchars($output);
            echo "</pre>";
            $vocab = json_decode($output, true);

            if (json_last_error() === JSON_ERROR_NONE) {
                $_SESSION['vocab_list'] = $vocab;
                header("Location: /lingoloop/view/select_vocab.php");

                exit();
            } else {
                echo "Error decoding vocabulary list.";
            }
        }
    }
}
```



```
}

return "Invalid credentials.";
}

public function register(string $username, string $email, string $password): ?string {
    if ($this->userModel->exists($username)) {
        return "Username already taken.";
    }

    $created = $this->userModel->create($username, $email, $password);

    if ($created) {
        // Automatski login i redirect na profil setup
        $user = $this->userModel->findByUsername($username);
        SessionManager::start($user['id'], $user['username']);
        header("Location: /lingoloop/view/setup_profile.php");
        exit();
    }

    return "Registration failed. Try again.";
}
}
```

LogoutController.php

```
<?php  
define('BASE_PATH', dirname(__DIR__));  
require_once BASE_PATH . '/models/SessionManager.php';
```

```
SessionManager::destroy();
```

```
header("Location: /lingoloop/view/?action=login");
```

```
exit();
```

SaveProfileController.php

```
<?php
require_once __DIR__ . '/../core/Database.php';
require_once __DIR__ . '/../core/SessionManager.php';

SessionManager::startSession();
if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/public/?action=login");
    exit();
}

$db = Database::getInstance();
$user_id = $_POST['user_id'] ?? null;

// Form data
$first_name = $_POST['first_name'] ?? '';
$last_name = $_POST['last_name'] ?? '';
$birth_date = $_POST['birth_date'] ?? '';
$country = $_POST['country'] ?? '';
$english_level = $_POST['english_level'] ?? '';
$learning_goal = $_POST['learning_goal'] ?? '';
$learning_time_per_day = $_POST['learning_time_per_day'] ?? '';
$learning_style = $_POST['learning_style'] ?? '';
$previous_apps = $_POST['previous_apps'] ?? '';
$interests = $_POST['interests'] ?? '';
$favorite_content = $_POST['favorite_content'] ?? '';

// Save profile
$sql = "INSERT INTO user_profiles
    (user_id, first_name, last_name, birth_date, country, english_level,
    learning_goal, learning_time_per_day, learning_style, previous_apps,
    interests, favorite_content)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

$stmt = $db->prepare($sql);
if (!$stmt) {
    die("Prepare failed: " . $db->error);
}

$stmt->bind_param(
    "issssssssss",
    $user_id,
    $first_name,
    $last_name,
    $birth_date,
    $country,
    $english_level,
    $learning_goal,
    $learning_time_per_day,
    $learning_style,
    $previous_apps,
    $interests,
```

```
        $favorite_content
    );

    if ($stmt->execute()) {
        // Pozovi Python skriptu
        $command = "python ../app/models/ai_vocabulary_generation.py {$user_id} 2>&1";
        $output = shell_exec($command);
        $vocab = json_decode($output, true);

        if (json_last_error() === JSON_ERROR_NONE) {
            $_SESSION['vocab_list'] = $vocab;
            header("Location: /lingoloop/view/index.php?action=select_vocab");
        } else {
            echo "Error decoding vocabulary list.";
        }

        } else {
            echo "Error saving profile: " . $stmt->error;
        }
    }
    ?>
```

SaveVocabController.php

```
<?php
require_once BASE_PATH . '/core/SessionManager.php';
require_once BASE_PATH . '/core/Database.php';

SessionManager::startSession();

if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/index.php?action=login");

    exit();
}

$userId = $_SESSION['user_id'] ?? null;
$vocabList = $_SESSION['vocab_list'] ?? null;
$selectedIndexes = json_decode($_POST['selected_words'] ?? '[]');
echo $userId;
echo "2232323223";

$db = Database::getInstance();
$stmt = $db->prepare("INSERT INTO user_vocabulary
    (user_id, term, translation, date_added, last_used, points, next_review_date)
    VALUES (?, ?, ?, NOW(), NULL, 0, NOW())
");

foreach ($selectedIndexes as $index) {
    if (!isset($vocabList[$index])) continue;

    $term = $vocabList[$index][0];
    $translation = $vocabList[$index][1];

    $stmt->bind_param("iss", $userId, $term, $translation);
    $stmt->execute();
}

// Očistimo sesiju nakon unosa
unset($_SESSION['vocab_list']);

header("Location: /lingoloop/view/index.php?action=welcome");

exit();
```

ai_title_generation.py

```
import sys
import os
import json
import ast
import time # dodaj import

# Podesi putanju do core/
sys.path.append("C:/xampp/htdocs/LingoLoop")
from models.Database import *
from groq import Groq

class AI_VOCABULARY_GENERATION:
    def __init__(self):
        self.__connection = Database.get_instance()
        self.__cursor = self.__connection.cursor()
        config_path = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'config', 'keys.json'))
        with open(config_path, 'r') as file:

            data = json.load(file)
            groq = data["Groq"]
            self.__client = Groq(api_key=groq)

    def getting_data_from_ab(self, id):
        query = "SELECT term FROM user_vocabulary WHERE user_id = %s AND next_review_date <= CURDATE() "
        self.__cursor.execute(query, (id,))
        column_names = [desc[0] for desc in self.__cursor.description]
        rezultati = self.__cursor.fetchall()

        output = ""
        for red in rezultati:
            for col_name, value in zip(column_names, red):
                output += f"{col_name}: {value}\n"
        return output

    def generate_youtube_titles(self, vocab_list):
        prompt = f"""
        You are a creative English teacher and YouTube content creator.
```

Here is a list of English vocabulary or idiomatic expressions that a language learner wants to study:

{vocab_list}

Based on this list, generate exactly **3** YouTube video titles in **English**.

Each title should:

- Be short (just a few words, not full sentences)
- Be catchy and creative
- Be directly related to the vocabulary
- Sound like a real YouTube video someone would want to click
- Be helpful for someone learning English

△ Return the result as a valid Python list of 3 strings. No explanations. No bullet points. Just something like:

```
["Title one", "Title two", "Title three"]
```

```
"""
    completion = self.__client.chat.completions.create(
        model="llama-3.3-70b-versatile",
        messages=[
            {"role": "system", "content": "You are a creative and engaging English language YouTube content creator."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.9,
        max_tokens=100,
        top_p=1
    )
    response = completion.choices[0].message.content.strip()

    try:
        # Parse the result into a real Python list
        titles = ast.literal_eval(response)
        return titles
    except Exception as e:
        # fallback in case parsing fails
        return []

# CLI poziv iz PHP-a
if __name__ == "__main__":
    x = AI_VOCABULARY_GENERATION()
    z = x.getting_data_from_ab(1)
    k = x.generate_youtube_titles(z)
    print(k)

#print(json.dumps(result))# Final output
```

ai_vocabulary_generation.py

```
import sys
import os
import json
import ast
import time # dodaj import

# Podesi putanju do core/
sys.path.append("C:/xampp/htdocs/LingoLoop")
from models.Database import *
from groq import Groq

class AI_VOCABULARY_GENERATION:
    def __init__(self):
        self.__connection = Database.get_instance()
        self.__cursor = self.__connection.cursor()
        config_path = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'config', 'keys.json'))
        with open(config_path, 'r') as file:

            data = json.load(file)
            groq = data["Groq"]
            self.__client = Groq(api_key=groq)

    def getting_data_from_ab(self, id):
        query = "SELECT * FROM user_profiles WHERE user_id = %s"
        self.__cursor.execute(query, (id,))
        column_names = [desc[0] for desc in self.__cursor.description]
        rezultati = self.__cursor.fetchall()

        output = ""
        for red in rezultati:
            for col_name, value in zip(column_names, red):
                output += f"{col_name}: {value}\n"
        return output

    def create_vocab(self, text):
        completion = self.__client.chat.completions.create(
            model="llama-3.3-70b-versatile",
            messages=[
                {"role": "system", "content": "You are a professional teacher fluent in French."},
                {"role": "user", "content": (
                    "Hier sind detaillierte Informationen über einen Benutzer:\n\n"
                    f"{text}\n\n"
                    "Bitte generiere eine Liste von **15 anspruchsvollen englischen Vokabeln oder idiomatischen Ausdrücken**, "
                    "die besonders gut zu diesem Benutzerprofil passen.\n\n"
                    "Beziehe dich auf seine Interessen, Ziele und seinen Lernstil, um relevante Ausdrücke zu wählen.\n\n"
                    "**Gib für jeden Begriff Folgendes an:**\n\n"
                    "1. Der englische Begriff\n\n"
                    "2. Die passende deutsche Übersetzung *(mit Artikel bei Substantiven)*\n\n"
                    "3. Einen kurzen Beispielsatz auf Englisch\n\n"
                    "Format:\n\n"
                    "[\n"
```



```

" ('term1', 'Übersetzung1'),\n"
" ('term2', 'Übersetzung2'),\n"
" ...\n"
"]\n\n"

```

"Gib **nur** die formatierte Python-kompatible Liste zurück – ohne zusätzliche Kommentare oder Erklärungen."

```

)}
],
temperature=1,
max_tokens=1024,
top_p=1,
stream=True,
stop=None,
)

```

```
response_text = "".join(chunk.choices[0].delta.content or "" for chunk in completion)
```

```

try:
    word_list = ast.literal_eval(response_text.strip())
    return word_list
except Exception as e:
    return []

```

CLI poziv iz PHP-a

```

if __name__ == "__main__":
    user_id = int(sys.argv[1])
    x = AI_VOCABULARY_GENERATION()
    z = x.getting_data_from_ab(user_id)
    y = x.create_vocab(z)

```

```
result = y
```

```
print(json.dumps(result))# Final output
```

Database.php

```
<?php

/**
 * Class Database
 *
 * Manages a single instance of MySQL database connection using Singleton pattern.
 */
class Database {
    private static ?mysqli $instance = null;

    /**
     * getInstance
     *
     * Returns the shared mysqli connection instance.
     *
     * @return mysqli
     */
    public static function getInstance(): mysqli {
        if (self::$instance === null) {
            require __DIR__ . '/../config/config.php';

            self::$instance = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);

            if (self::$instance->connect_error) {
                die("Database connection failed: " . self::$instance->connect_error);
            }
        }

        return self::$instance;
    }

    /**
     * Private constructor to prevent instantiation.
     */
    private function __construct() {}

    /**
     * Private clone method to prevent cloning the instance.
     */
    private function __clone() {}

    /**
     * Private unserialize method to prevent restoring from string.
     */
    public function __wakeup() {}
}
```

Database.py

```
import mysql.connector
from config.config import DB_CONFIG

class Database:
    __instance = None # klasna promenljiva

    @classmethod
    def get_instance(cls):

        if cls.__instance is None:
            cls.__instance = mysql.connector.connect(
                host=DB_CONFIG['host'],
                user=DB_CONFIG['user'],
                password=DB_CONFIG['password'],
                database=DB_CONFIG['database']
            )
        return cls.__instance

    @classmethod
    def close_connection(cls):
        if cls.__instance is not None:
            cls.__instance.close()
            cls.__instance = None
```

SaveProfile.php

```
<?php
```

```
class SaveProfile
```

```
{
```

```
    private \mysqli $db;
```

```
    public function __construct(mysqli $db)
```

```
    {
```

```
        $this->db = $db;
```

```
    }
```

```
    public function create(array $data): bool
```

```
    {
```

```
        $sql = "INSERT INTO user_profiles (  
            user_id, first_name, last_name, birth_date, country,  
            english_level, learning_goal, learning_time_per_day,  
            learning_style, previous_apps, interests, favorite_content  
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

```
        $stmt = $this->db->prepare($sql);
```

```
        if (!$stmt) {
```

```
            throw new Exception("Prepare failed: " . $this->db->error);
```

```
        }
```

```
        $stmt->bind_param(  
            "issssssssss",
```

```
            $data['user_id'],  
            $data['first_name'],
```

```
            $data['last_name'],
```

```
            $data['birth_date'],
```

```
            $data['country'],
```

```
            $data['english_level'],
```

```
            $data['learning_goal'],
```

```
            $data['learning_time_per_day'],
```

```
            $data['learning_style'],
```

```
            $data['previous_apps'],
```

```
            $data['interests'],
```

```
            $data['favorite_content']
```

```
        );
```

```
        return $stmt->execute();
```

```
    }
```

```
}
```

SaveVocab.php

<?php

```
class UserVocabulary
{
    private \mysqli $db;

    public function __construct(mysqli $db)
    {
        $this->db = $db;
    }

    /**
     * Sprema selektovane reči korisnika u bazu.
     *
     * @param int $userId
     * @param array $vocabList - cela lista reči iz sesije (index => [term, translation])
     * @param array $selectedIndexes - niz indeksa koje je korisnik odabrao
     * @return void
     */
    public function saveSelectedWords(int $userId, array $vocabList, array $selectedIndexes): void
    {
        $stmt = $this->db->prepare("INSERT INTO user_vocabulary
            (user_id, term, translation, date_added, last_used, points, next_review_date)
            VALUES (?, ?, ?, NOW(), NULL, 0, NOW())
        ");

        if (!$stmt) {
            throw new Exception("Database error: " . $this->db->error);
        }

        foreach ($selectedIndexes as $index) {
            if (!isset($vocabList[$index])) continue;

            $term = $vocabList[$index][0];
            $translation = $vocabList[$index][1];

            $stmt->bind_param("iss", $userId, $term, $translation);
            $stmt->execute();
        }

        $stmt->close();
    }

    public function hasAnyWords(int $userId): bool
    {
        $query = "SELECT 1 FROM user_vocabulary WHERE user_id = ? LIMIT 1";
        $stmt = $this->db->prepare($query);
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $stmt->store_result();
    }
}
```

```
    return $stmt->num_rows > 0;
}
}
```

SessionManager.php

```
<?php

/**
 * Class SessionManager
 *
 * Manages user sessions: start, destroy, and check login status.
 */
class SessionManager {

    /**
     * start
     *
     * Starts a new session and stores user data.
     *
     * @param int $userId
     * @param string $username
     * @return void
     */
    public static function start(int $userId, string $username): void {
        if (session_status() === PHP_SESSION_NONE) {
            session_start();

            $_SESSION['user_id'] = $userId;
            $_SESSION['username'] = $username;
        }

        /**
         * destroy
         *
         * Ends the session and clears session data.
         *
         * @return void
         */
        public static function destroy(): void {
            if (session_status() === PHP_SESSION_NONE) {
                session_start();

                $_SESSION = [];
                session_destroy();
            }

            /**
             * isLoggedIn
             *
             * Checks if the user is currently logged in.
             *
             * @return bool
             */
            public static function isLoggedIn(): bool {
```

```
    return isset($_SESSION['user_id']);  
}
```

```
public static function startSession(): void {  
    if (session_status() === PHP_SESSION_NONE) {  
        session_start();  
    }  
}
```

```
/**  
 * getUsername  
 *  
 * Returns the currently logged-in username.  
 *  
 * @return string|null  
 */  
public static function getUsername(): ?string {  
    return $_SESSION['username'] ?? null;  
}
```

```
}
```


User.php

```
<?php

/**
 * Class User
 *
 * Handles user-related database operations such as finding users,
 * creating new accounts, and checking for existing usernames.
 */
class User {
    private $db;
    private $table = 'users';

    /**
     * Constructor
     *
     * @param mysqli $db - MySQLi database connection object
     */
    public function __construct(mysqli $db) {
        $this->db = $db;
    }

    /**
     * findByUsername
     *
     * Finds a user by their username.
     *
     * @param string $username
     * @return array|null - Returns user data as an associative array, or null if not found
     */
    public function findByUsername(string $username): ?array {
        $sql = "SELECT * FROM {$this->table} WHERE username = ?";
        $stmt = $this->db->prepare($sql);
        $stmt->bind_param('s', $username);
        $stmt->execute();
        $result = $stmt->get_result();

        return $result->fetch_assoc() ?: null;
    }

    /**
     * create
     *
     * Creates a new user in the database.
     *
     * @param string $username - The desired username
     * @param string $email - User's email address
     * @param string $password - Raw password (will be hashed)
     * @return bool - Returns true if user was created successfully
     */
    public function create(string $username, string $email, string $password): bool {
        $hashedPassword = password_hash($password, PASSWORD_BCRYPT);
```

```

        $sql = "INSERT INTO {$this->table} (username, email, password) VALUES (?, ?, ?)";
        $stmt = $this->db->prepare($sql);
        $stmt->bind_param('sss', $username, $email, $hashedPassword);

        return $stmt->execute();
    }

    /**
     * exists
     *
     * Checks whether a user with the given username already exists.
     *
     * @param string $username
     * @return bool - Returns true if the user exists
     */
    public function exists(string $username): bool {
        return $this->findByUsername($username) !== null;
    }

    public function hasProfile(int $userId): bool {
        $sql = "SELECT 1 FROM user_profiles WHERE user_id = ?";
        $stmt = $this->db->prepare($sql);
        $stmt->bind_param('i', $userId);
        $stmt->execute();
        $stmt->store_result();

        return $stmt->num_rows > 0;
    }
}

```

youtube.py

```
from googleapiclient.discovery import build
from datetime import timedelta
from ai_title_generation import *
import isodate
import sys
import os
import json
import ast
from youtube_transcript_api import YouTubeTranscriptApi
from youtube_transcript_api._errors import TranscriptsDisabled, NoTranscriptAvailable
from groq import Groq
import textwrap

sys.path.append("C:/xampp/htdocs/LingoLoop")
sys.stdout.reconfigure(encoding='utf-8')
```

```
class YOUTUBE:
```

```
    def __init__(self):
        config_path = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'config', 'keys.json'))
        with open(config_path, 'r') as file:
            data = json.load(file)
            self.__API_key = data["Google"]
            groq_key = data["Groq"]
```

```
    self.__title = AI_VOCABULARY_GENERATION()
    self.__client = Groq(api_key=groq_key)
```

```
    def get_tittles(self, user_id):
        user_data = self.__title.getting_data_from_ab(user_id)
        titles = self.__title.generate_youtube_titles(user_data)
        return titles, user_data
```

```
    def search_youtube_videos(self, query, max_results=50, min_views=50000, return_limit=3):
        youtube = build("youtube", "v3", developerKey=self.__API_key)
        search_response = youtube.search().list(
            part="id",
            q=query,
            type="video",
            maxResults=max_results,
            videoDuration="medium"
        ).execute()
```

```
        video_ids = [item["id"]["videoid"] for item in search_response["items"]]
```

```
        videos_response = youtube.videos().list(
            part="snippet,contentDetails,statistics",
            id=", ".join(video_ids)
        ).execute()
```

```
        filtered_videos = []
```

```

for item in videos_response["items"]:
    try:
        duration_iso = item["contentDetails"]["duration"]
        duration = isodate.parse_duration(duration_iso)
        view_count = int(item["statistics"]["viewCount"])

        if timedelta(minutes=8) <= duration <= timedelta(minutes=15) and view_count >= min_views:
            video_data = {
                "title": item["snippet"]["title"],
                "url": f"https://www.youtube.com/watch?v={item['id']}",
                "duration": str(duration),
                "views": view_count
            }
            filtered_videos.append(video_data)
            if len(filtered_videos) >= return_limit:
                break
    except Exception:
        continue

return filtered_videos

```

```

def get_transcript_en(self, video_url):
    try:
        video_id = video_url.split("v=")[-1]
        transcript = YouTubeTranscriptApi.get_transcript(video_id, languages=['en'])
        text = " ".join([entry["text"] for entry in transcript])
        return text
    except (TranscriptsDisabled, NoTranscriptAvailable):
        return "[Transcript not available]"
    except Exception as e:
        return f"[Error: {str(e)}]"

```

```

def find_top_videos(self, vocab_list, video_data_list, top_n=4):
    # Ukloni duplikate na osnovu URL-a
    unique_videos = {video["url"]: video for video in video_data_list}.values()

    for video in unique_videos:
        video["transcript"] = video["transcript"][:1000]

    prompt = f"""

```

You are an expert English tutor helping students find the best YouTube videos for studying vocabulary.

The student wants to learn these words and expressions:
{vocab_list}

Here is a list of YouTube videos with titles, URLs, and transcripts. From this list, select the ****TOP {top_n}**** videos that are most useful for learning these words.

Prefer videos where the vocabulary appears in the transcript, or the content is closely related.

Return a valid Python list of dictionaries like:

```

[
    {"title": "...", "url": "..."},

```

...

]

No explanations, no bullet points.

Videos:

```
{json.dumps(list(unique_videos), ensure_ascii=False, indent=2)}
```

"""

```
completion = self.__client.chat.completions.create(
    model="llama3-8b-8192",
    messages=[
        {"role": "system", "content": "You are a helpful English language tutor and recommender."},
        {"role": "user", "content": prompt}
    ],
    temperature=0.7,
    max_tokens=900,
    top_p=1
)
try:
    return ast.literal_eval(completion.choices[0].message.content.strip())
except Exception:
    return []
```

```
def generate_short_description(self, transcript):
    prompt = f"""
```

You are a helpful assistant. Summarize the following transcript into a **very short YouTube description** that is:

- Maximum 50 characters
- Simple and clear
- Describes what the video is about

Transcript:

```
\"\"\"
{transcript}
\"\"\"
```

Return only the description string, no bullet points, no extra formatting.

"""

```
try:
    completion = self.__client.chat.completions.create(
        model="llama3-8b-8192",
        messages=[
            {"role": "system", "content": "You are a concise YouTube video summarizer."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.7,
        max_tokens=50,
        top_p=1
    )
    return completion.choices[0].message.content.strip()
except Exception:
    return "No description generated"
```

```
def display_video_summary(self, video_url, transcript):
    youtube = build("youtube", "v3", developerKey=self.__API_key)
```

```

video_id = video_url.split("v=")[-1]

response = youtube.videos().list(
    part="snippet,contentDetails",
    id=video_id
).execute()

if not response["items"]:
    return

item = response["items"][0]
title = item["snippet"]["title"]
description = self.generate_short_description(transcript)
duration = isodate.parse_duration(item["contentDetails"]["duration"])

return [title,description,str(duration),f'https://www.youtube.com/watch?v={video_id}']

def fetch_top_video_summaries(self, user_id, max_videos=10, top_n=4):
    queries, words = self.get_tittles(user_id)

    all_video_data = []
    seen_urls = set()

    for query in queries:
        results = self.search_youtube_videos(query)
        for video in results:
            if video["url"] not in seen_urls:
                seen_urls.add(video["url"])
                transcript = self.get_transcript_en(video["url"])
                short_transcript = transcript[:1000]
                all_video_data.append({
                    "title": video["title"],
                    "url": video["url"],
                    "transcript": short_transcript
                })
            if len(all_video_data) >= max_videos:
                break

    top_videos = self.find_top_videos(words, all_video_data, top_n=top_n)
    summaries = []

    for video in top_videos:
        match = next((v for v in all_video_data if v["url"] == video["url"]), None)
        if match:
            summary = self.display_video_summary(video["url"], match["transcript"])
            if summary:
                summaries.append(summary)

    return summaries

def convert_transcript_to_readable_text(self, video_url):
    transcript = self.get_transcript_en(video_url)

```

```
if not transcript or transcript.startswith("["):
    return "Transcript not available or could not be retrieved."
```

```
chunks = textwrap.wrap(transcript, width=3500) # oko 700-900 tokena po chunku
full_output = ""
```

```
for i, chunk in enumerate(chunks):
    prompt = f"""
```

You are a helpful assistant. The following is a raw transcript from a YouTube video.
Your task is to lightly edit it so that it becomes a clean, well-structured, readable article.

⚠ IMPORTANT:

- Do NOT change or remove facts, names, or events.
- Do NOT invent or add content.
- Keep the original sequence and meaning.
- Your goal is only to improve grammar, punctuation, and structure for easier reading.

Transcript Part {i+1}:

```
\"\"\"{chunk}\"\"\"
```

Now rewrite this part as a readable article with paragraphs. Return only the text.

```
"""
```

```
try:
    completion = self.__client.chat.completions.create(
        model="llama3-8b-8192",
        messages=[
            {"role": "system", "content": "You are an assistant that improves transcript readability without changing the content."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.3,
        max_tokens=1800,
        top_p=1
    )
    part_output = completion.choices[0].message.content.strip()
    full_output += part_output + "\n\n"
except Exception as e:
    full_output += f"[Error generating part {i+1}: {str(e)}]\n\n"

return full_output.strip()
```

```
if __name__ == "__main__":
    yt = YOUTUBE()
```

```
if len(sys.argv) == 2:
    # Samo jedan argument
    param = sys.argv[1]
```

```
if param.startswith("http"):
    # Radi se o video URL-u
    result = yt.convert_transcript_to_readable_text(param)
    print(result)
else:
    # Pretpostavljamo da je user_id
    result = yt.fetch_top_video_summaries(int(param), max_videos=10, top_n=4)
    print(json.dumps(result))

elif len(sys.argv) == 3 and sys.argv[2] == "transcript":
    video_url = sys.argv[1]
    result = yt.convert_transcript_to_readable_text(video_url)
    print(result)

else:
    print("[Error: Invalid arguments provided.]")
```


dashboard.php

```
<?php
define('BASE_PATH', dirname(__DIR__));
require_once BASE_PATH . '/models/SessionManager.php';

SessionManager::startSession();

if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/login.php");
    exit();
}

$username = $_SESSION['username'];
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>LingoLoop | Dashboard</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            background-color: #000;
            color: #fff;
            font-family: Arial, sans-serif;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            padding: 20px;
        }

        h1 {
            font-size: 2rem;
            margin-bottom: 20px;
        }

        .btn-container {
            display: flex;
            flex-direction: column;
            gap: 15px;
            width: 100%;
            max-width: 300px;
        }

        .btn {
            background-color: #1e1e1e;
            color: #fff;
            padding: 15px;
            font-size: 1.2rem;
        }
```

```
font-weight: bold;
border: none;
border-radius: 8px;
cursor: pointer;
transition: background-color 0.3s ease;
text-align: center;
text-decoration: none;
}

.btn:hover {
    background-color: #007bff;
}

a.logout {
    margin-top: 30px;
    color: #aaa;
    text-decoration: none;
    font-size: 1rem;
}

a.logout:hover {
    color: #fff;
}
</style>
</head>
<body>

<h1>Welcome back, <?= htmlspecialchars($username) ?> </h1>

<div class="btn-container">
    <a href="/lingoloop/view/select_vocab.php" class="btn"> Vocabulary</a>
    <a href="/lingoloop/view/watch_video.php" class="btn"> Watch Videos</a>
    <a href="/lingoloop/view/ichat.php" class="btn"> iChatting</a>
</div>

<a href="/lingoloop/controller/LogoutController.php" class="logout">Log out</a>

</body>
</html>
```

index.php

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8">
  <title>Weiterleitung...</title>
  <meta http-equiv="refresh" content="0; URL=login.php">
</head>
<body>
  <p>Du wirst weitergeleitet nach <a href="login.php">login.php</a>...</p>
</body>
</html>
```

login.php

```
<?php
define('BASE_PATH', dirname(__DIR__));

require_once __DIR__ . '/../models/SessionManager.php';
require_once __DIR__ . '/../controller/AuthController.php';
$auth = new AuthController();
if (SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/index.php?action=welcome");
    exit();
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $auth = new AuthController();
    $username = $_POST['username'] ?? '';
    $password = $_POST['password'] ?? '';

    $error = $auth->login($username, $password);
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login | LingoLoop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        /* Reset */
        * { margin: 0; padding: 0; box-sizing: border-box; }
        body {
            background-color: #000;
            color: #fff;
            font-family: Arial, sans-serif;
            display: flex;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            padding: 20px;
        }
        .container {
            background-color: #111;
            padding: 40px;
            border-radius: 8px;
            width: 100%;
            max-width: 600px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);
            animation: fadeIn 1s ease-in-out;
        }
        @keyframes fadeIn {
            from { opacity: 0; }

```

```
    to { opacity: 1; }
}
h2 {
  font-size: 2rem;
  font-weight: bold;
  text-align: center;
  margin-bottom: 20px;
}
.form-group {
  margin-bottom: 20px;
}
label {
  display: block;
  font-size: 1rem;
  margin-bottom: 5px;
}
input[type="text"],
input[type="password"] {
  width: 100%;
  padding: 12px;
  border: 1px solid #333;
  border-radius: 4px;
  background-color: #222;
  color: #fff;
  font-size: 1rem;
}
button {
  width: 100%;
  padding: 15px;
  border: none;
  border-radius: 4px;
  background-color: #007BFF;
  color: #fff;
  font-size: 1.2rem;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
button:hover {
  background-color: #0056b3;
}
p {
  font-size: 1rem;
  text-align: center;
  margin-top: 20px;
  color: #aaa;
}
a {
  color: #00f;
  text-decoration: none;
}
a:hover {
  text-decoration: underline;
```

```
}
@media (max-width: 600px) {
  .container {
    padding: 20px;
  }
  h2 {
    font-size: 1.5rem;
  }
  button {
    font-size: 1rem;
  }
}
</style>
</head>
<body>
<div class="container">
  <h2>Login to LingoLoop</h2>
  <?php if (isset($error)) echo "<p style='color:red;'>$error</p>"; ?>
  <form method="POST" action="login.php">
    <input type="hidden" name="action" value="login">
    <div class="form-group">
      <label>Username</label>
      <input type="text" name="username" placeholder="Enter your username" required>
    </div>
    <div class="form-group">
      <label>Password</label>
      <input type="password" name="password" placeholder="Enter your password" required>
    </div>
    <button type="submit">Login</button>
  </form>
  <p>Don't have an account? <a href="/lingoloop/view/register.php">Register here</a></p>
</div>
</body>
</html>
```

register.php

```
<?php
define('BASE_PATH', dirname(__DIR__));
require_once BASE_PATH . '/models/SessionManager.php';
require_once __DIR__ . '/../controller/AuthController.php';

if (SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/index.php?action=welcome");
    exit();
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $auth = new AuthController();
    $username = trim($_POST['username']);
    $email = trim($_POST['email']);
    $password = trim($_POST['password']);
    $_SESSION['raw_password'] = $password;
    $error = $auth->register($username, $email, $password);
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Register | LingoLoop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        /* Reset */
        * { margin: 0; padding: 0; box-sizing: border-box; }
        body {
            background-color: #000;
            color: #fff;
            font-family: Arial, sans-serif;
            display: flex;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            padding: 20px;
        }
        .container {
            background-color: #111;
            padding: 40px;
            border-radius: 8px;
            width: 100%;
            max-width: 600px;
            box-shadow: 0 4px 8px rgba(0,0,0,0.5);
            animation: fadeIn 1s ease-in-out;
        }
        @keyframes fadeIn {
            from { opacity: 0; }
```

```
    to { opacity: 1; }
}
h2 {
    font-size: 2rem;
    font-weight: bold;
    text-align: center;
    margin-bottom: 20px;
}
.form-group {
    margin-bottom: 20px;
}
label {
    display: block;
    font-size: 1rem;
    margin-bottom: 5px;
}
input[type="text"],
input[type="email"],
input[type="password"] {
    width: 100%;
    padding: 12px;
    border: 1px solid #333;
    border-radius: 4px;
    background-color: #222;
    color: #fff;
    font-size: 1rem;
}
button {
    width: 100%;
    padding: 15px;
    border: none;
    border-radius: 4px;
    background-color: #28a745; /* zelena boja */
    color: #fff;
    font-size: 1.2rem;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
button:hover {
    background-color: #218838;
}
p {
    font-size: 1rem;
    text-align: center;
    margin-top: 20px;
    color: #aaa;
}
a {
    color: #00f;
    text-decoration: none;
}
a:hover {
```



```
text-decoration: underline;
}
@media (max-width: 600px) {
  .container {
    padding: 20px;
  }
  h2 {
    font-size: 1.5rem;
  }
  button {
    font-size: 1rem;
  }
}
</style>
</head>
<body>
<div class="container">
  <h2>Create an Account</h2>
  <?php if (isset($error)) echo "<p style='color:red;'>$error</p>"; ?>
  <form method="POST" action="/lingoloop/view/register.php">
    <input type="hidden" name="action" value="register">
    <div class="form-group">
      <label>Username</label>
      <input type="text" name="username" placeholder="Enter your username" required>
    </div>
    <div class="form-group">
      <label>Email</label>
      <input type="email" name="email" placeholder="Enter your email" required>
    </div>
    <div class="form-group">
      <label>Password</label>
      <input type="password" name="password" placeholder="Enter your password" required>
    </div>
    <button type="submit">Register</button>
  </form>
  <p>Already have an account? <a href="/lingoloop/view/login.php">Login here</a></p>
</div>
</body>
</html>
```

select_video.php

```
<?php
// select_video.php

ini_set('display_errors', 1);
error_reporting(E_ALL);

$videoid = $_GET['video_id'] ?? null;

if (!$videoid) {
    echo "<h2> Kein Video ausgewählt. Bitte gehe zurück und wähle ein Video.</h2>";
    exit();
}

$videoUrl = "https://www.youtube.com/watch?v=$videoid";
?>

<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Was willst du tun?</title>
    <style>
        body {
            background-color: #121212;
            color: #fff;
            font-family: 'Segoe UI', sans-serif;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            text-align: center;
        }
        h1 {
            font-size: 2rem;
            margin-bottom: 30px;
        }
        .button-container {
            display: flex;
            flex-direction: column;
            gap: 20px;
            width: 90%;
            max-width: 400px;
        }
        a.button {
            background-color: #007bff;
            color: white;
            padding: 15px;
            border-radius: 8px;
            text-decoration: none;
            font-size: 1.2rem;
        }
    </style>
</head>
<body>
    <h1>Was willst du tun?</h1>
    <div class="button-container">
        <a href="$videoUrl">Wähle ein Video</a>
    </div>
</body>
</html>
```

```
        font-weight: bold;
        transition: background-color 0.3s ease;
    }
    a.button:hover {
        background-color: #0056b3;
    }
</style>
</head>
<body>

<h1>Was willst du zuerst machen?</h1>
<div class="button-container">
    <a class="button" href="/lingoloop/view/show_transcript.php?video_id=<?= urlencode($videoid) ?>"> Transkript
anzeigen</a>
    <a class="button" href="<?= htmlspecialchars($videoUrl) ?>" target="_blank">► Direkt zum Video</a>
</div>

</body>
</html>
```

select_vocab.php

```
<?php
define('BASE_PATH', dirname(__DIR__)); // falls notwendig anpassen

require_once BASE_PATH . '/models/SessionManager.php';
require_once BASE_PATH . '/models/Database.php';
require_once BASE_PATH . '/models/SaveVocab.php';

SessionManager::startSession();

if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/index.php?action=login");
    exit();
}

$userId = $_SESSION['user_id'] ?? null;
$vocabList = $_SESSION['vocab_list'] ?? [];
$selectedIndexes = json_decode($_POST['selected_words'] ?? '[]', true);

$db = Database::getInstance();
$vocabHandler = new UserVocabulary($db);

if($vocabHandler->hasAnyWords($userId)){
    header("Location: /lingoloop/view/dashboard.php");
    exit();
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    try {
        $vocabHandler->saveSelectedWords($userId, $vocabList, $selectedIndexes);
        // Nach dem Speichern löschen wir die Liste aus der Session
        unset($_SESSION['vocab_list']);
        header("Location: /lingoloop/view/dashboard.php");
        exit();
    } catch (Exception $e) {
        echo "Fehler: " . $e->getMessage();
    }
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Select Your Favorite Vocabulary</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script>
        let selectedWords = [];

        function toggleSelection(index, button) {
            const maxSelection = 10;
            const word = index.toString();
```

```

const alreadySelected = selectedWords.includes(word);

if (alreadySelected) {
    selectedWords = selectedWords.filter(w => w !== word);
    button.classList.remove('selected');
} else {
    if (selectedWords.length >= maxSelection) {
        alert("You can only select up to 10 words.");
        return;
    }
    selectedWords.push(word);
    button.classList.add('selected');
}

document.getElementById('selectedWords').value = JSON.stringify(selectedWords);
}
</script>
<style>
    body { background-color: #000; color: #fff; font-family: Arial; padding: 20px; }
    .word-box { background-color: #111; margin-bottom: 10px; padding: 15px; border-radius: 8px; display: flex; justify-content:
space-between; align-items: center; }
    .heart-btn { cursor: pointer; font-size: 24px; }
    .selected { color: red; }
    button[type="submit"] { padding: 10px 20px; margin-top: 20px; border: none; background-color: #28a745; color: white;
font-size: 1.1rem; border-radius: 4px; cursor: pointer; }
</style>
</head>
<body>

<h2>Select Up to 10 Favorite Words ♥</h2>

<form method="POST" action="/lingoloop/view/select_vocab.php">
    <?php foreach ($vocabList as $index => $item): ?>
        <div class="word-box">
            <div>
                <strong><?= htmlspecialchars($item[0]) ?></strong> - <?= htmlspecialchars($item[1]) ?>
            </div>
            <div class="heart-btn" onclick="toggleSelection(<?= $index ?>, this)">&#10084;</div>
        </div>
    <?php endforeach; ?>

    <input type="hidden" name="selected_words" id="selectedWords" value="[]">
    <button type="submit">Save Selected Words</button>
</form>

</body>
</html>

```

setup_profile.php

```
<?php
define('BASE_PATH', dirname(__DIR__));

require_once BASE_PATH . '/models/SessionManager.php';
require_once BASE_PATH . '/models/Database.php';
require_once BASE_PATH . '/models/SaveProfile.php';
require_once BASE_PATH . '/controller/AuthController.php';

SessionManager::startSession();

// Ako nije ulogovan → redirect
if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/login.php");
    exit();
}

$userId = $_SESSION['user_id'];
$db = Database::getInstance();
$saveProfile = new SaveProfile($db);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $data = [
        'user_id' => $userId,
        'first_name' => trim($_POST['first_name']),
        'last_name' => trim($_POST['last_name']),
        'birth_date' => trim($_POST['birth_date']),
        'country' => trim($_POST['country']),
        'english_level' => trim($_POST['english_level']),
        'learning_goal' => trim($_POST['learning_goal']),
        'learning_time_per_day' => trim($_POST['learning_time_per_day']),
        'learning_style' => trim($_POST['learning_style']),
        'previous_apps' => trim($_POST['previous_apps']),
        'interests' => trim($_POST['interests']),
        'favorite_content' => trim($_POST['favorite_content']),
    ];
    $saveProfile->create($data);
}

$username = $_SESSION['username'];
$userId = $_SESSION['user_id'];
?>

<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Profil-Einrichtung | LingoLoop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<!-- Koristimo Alpine.js samo za navigaciju wizardom -->
<script src="https://cdn.jsdelivr.net/npm/alpinejs" defer></script>
<style>
  /* Osnovni reset i stilovi */
  * { box-sizing: border-box; margin: 0; padding: 0; }
  body {
    background-color: #000;
    color: #fff;
    font-family: Arial, sans-serif;
    display: flex;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
    padding: 20px;
  }
  .container {
    background-color: #111;
    padding: 20px;
    border-radius: 8px;
    width: 100%;
    max-width: 600px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.5);
  }
  h2 { margin-bottom: 20px; text-align: center; }
  .step { display: none; opacity: 0; transition: opacity 0.5s ease-in-out; }
  .step.active { display: block; opacity: 1; }
  .form-group { margin-bottom: 15px; }
  label { display: block; margin-bottom: 5px; font-size: 0.9em; }
  input[type="text"],
  input[type="date"],
  select,
  textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #333;
    border-radius: 4px;
    background-color: #222;
    color: #fff;
    font-size: 0.95em;
  }
  textarea { resize: vertical; min-height: 80px; }
  .nav-buttons {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
  }
  .nav-buttons button {
    border: none;
    padding: 10px 20px;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
    color: #fff;
  }
```

```

}
#nextBtn { background-color: #28a745; } /* zelena */
#nextBtn:hover { background-color: #218838; }
#submitBtn { background-color: #dc3545; } /* crvena */
#submitBtn:hover { background-color: #c82333; }
#prevBtn { background-color: #333; }
#prevBtn:hover { background-color: #555; }
@media (max-width: 600px) {
    .nav-buttons { flex-direction: column; }
    .nav-buttons button { width: 100%; margin-bottom: 10px; }
}
</style>
</head>
<body>
<div class="container" x-data="wizard()">
    <form id="profileForm" action="/lingoloop/view/setup_profile.php" method="POST" onsubmit="return validateForm()">
        <input type="hidden" name="user_id" value="<?= $userId ?>">

        <!-- Step 0: Einführung -->
        <div class="step" id="step0">
            <h2>Willkommen! </h2>
            <p style="margin-bottom: 20px;">
                Um Ihnen das bestmögliche Lernerlebnis zu bieten, bitten wir Sie, einige kurze Fragen zu beantworten.
                Ihre Antworten helfen uns, den Unterricht genau auf Ihre Interessen und Bedürfnisse abzustimmen.
                Alle Daten sind vollständig privat und werden ausschließlich zur Personalisierung Ihres Lernens verwendet.
                Vielen Dank für Ihre Zusammenarbeit!
            </p>
        </div>

        <!-- Step 1: Persönliche Daten -->
        <div class="step" id="step1">
            <h2>Schritt 1: Persönliche Daten</h2>
            <div class="form-group">
                <label>Vorname (z.B. "Max")</label>
                <input type="text" name="first_name" placeholder="Max" required>
            </div>
            <div class="form-group">
                <label>Nachname (z.B. "Mustermann")</label>
                <input type="text" name="last_name" placeholder="Mustermann" required>
            </div>
            <div class="form-group">
                <label>Geburtsdatum</label>
                <input type="date" name="birth_date" required>
            </div>
            <div class="form-group">
                <label>Land</label>
                <select name="country" required>
                    <option value="">Bitte wählen</option>
                    <option value="Deutschland">Deutschland</option>
                    <option value="Österreich">Österreich</option>
                    <option value="Schweiz">Schweiz</option>
                    <option value="Andere">Andere</option>
                </select>
            </div>
        </div>
    </form>
</div>

```



```

</div>
<div class="form-group">
  <label>Englisch Niveau (A1-C2)</label>
  <select name="english_level" required>
    <option value="">Bitte wählen</option>
    <option value="A1">A1</option>
    <option value="A2">A2</option>
    <option value="B1">B1</option>
    <option value="B2">B2</option>
    <option value="C1">C1</option>
    <option value="C2">C2</option>
  </select>
</div>
</div>

```

<!-- Step 2: Lernziele -->

```

<div class="step" id="step2">
  <h2>Schritt 2: Lernziele</h2>
  <div class="form-group">
    <label>Warum lernen Sie Englisch? (z.B. "Für die Arbeit")</label>
    <input type="text" name="learning_goal" placeholder="Für die Arbeit" required>
  </div>
  <div class="form-group">
    <label>Wie viel Zeit pro Tag? (z.B. "20 Minuten")</label>
    <select name="learning_time_per_day" required>
      <option value="">Bitte wählen</option>
      <option value="10 Minuten">10 Minuten</option>
      <option value="20 Minuten">20 Minuten</option>
      <option value="30+ Minuten">30+ Minuten</option>
    </select>
  </div>
</div>

```

<!-- Step 3: Lernstil -->

```

<div class="step" id="step3">
  <h2>Schritt 3: Lernstil</h2>
  <div class="form-group">
    <label>Bevorzugte Lernmethode (z.B. "Vokabeln üben, Videos anschauen, Lesen, Schreiben")</label>
    <input type="text" name="learning_style" placeholder="Vokabeln üben, Videos anschauen, Lesen, Schreiben"
required>
  </div>
  <div class="form-group">
    <label>Apps, die Sie bisher genutzt haben (optional)</label>
    <input type="text" name="previous_apps" placeholder="z.B. Duolingo">
  </div>
</div>

```

<!-- Step 4: Interessen & Hobbys -->

```

<div class="step" id="step4">
  <h2>Schritt 4: Interessen & Hobbys</h2>
  <div class="form-group">
    <label>Ihre Interessen, Hobbys und Leidenschaften</label>
    <textarea name="interests" placeholder="z.B. Fußball, Lesen, Musik (mehrere mit Komma trennen)"

```

```

required></textarea>
    </div>
    <div class="form-group">
        <label>Was schauen oder hören Sie am liebsten?</label>
        <textarea name="favorite_content" placeholder="z.B. YouTube-Kanäle, Podcasts" required></textarea>
    </div>
</div>

<!-- Navigation Buttons -->
<div class="nav-buttons">
    <button type="button" id="prevBtn" onclick="prevStep()" disabled>Zurück</button>
    <button type="button" id="nextBtn" onclick="nextStep()">Weiter</button>
    <button type="submit" id="submitBtn" style="display: none;">Fertig</button>
</div>
</form>
</div>

<script>
    let currentStep = 0;
    const steps = document.querySelectorAll('.step');
    const prevBtn = document.getElementById('prevBtn');
    const nextBtn = document.getElementById('nextBtn');
    const submitBtn = document.getElementById('submitBtn');

    function showStep(index) {
        steps.forEach((step, i) => {
            step.classList.toggle('active', i === index);
        });
        prevBtn.disabled = (index === 0);
        if (index === steps.length - 1) {
            nextBtn.style.display = 'none';
            submitBtn.style.display = 'inline-block';
        } else {
            nextBtn.style.display = 'inline-block';
            submitBtn.style.display = 'none';
        }
    }

    function nextStep() {
        if (validateCurrentStep()) {
            if (currentStep < steps.length - 1) {
                currentStep++;
                showStep(currentStep);
            }
        }
    }

    function prevStep() {
        if (currentStep > 0) {
            currentStep--;
            showStep(currentStep);
        }
    }

```

```

// Custom validation for current step (skip intro step)
function validateCurrentStep() {
  if (currentStep === 0) return true;
  const currentFields = steps[currentStep].querySelectorAll('input[required], select[required], textarea[required]');
  for (const field of currentFields) {
    if (!field.value.trim()) {
      alert("Bitte füllen Sie das Feld '" + field.previousElementSibling.textContent.trim() + "' aus.");
      field.focus();
      return false;
    }
  }
  return true;
}

function validateForm() {
  // Validate all steps before submitting
  for (let i = 1; i < steps.length; i++) {
    const fields = steps[i].querySelectorAll('input[required], select[required], textarea[required]');
    for (const field of fields) {
      if (!field.value.trim()) {
        alert("Bitte füllen Sie das Feld '" + field.previousElementSibling.textContent.trim() + "' aus.");
        currentStep = i;
        showStep(currentStep);
        field.focus();
        return false;
      }
    }
  }
  return true;
}

// Initialize first step
showStep(currentStep);
</script>
</body>
</html>

```

show_transcript.php

```
<?php
// show_transcript.php

ini_set('display_errors', 1);
error_reporting(E_ALL);

$videoid = $_GET['video_id'] ?? null;

if (!$videoid) {
    echo "<h2> Kein Video gefunden.</h2>";
    exit();
}

$videoUrl = "https://www.youtube.com/watch?v=$videoid";
$pythonPath = "python";
$scriptPath = "C:/xampp/htdocs/LingoLoop/models/youtube.py";
$command = escapeshellcmd("$pythonPath \"$scriptPath\" \"$videoUrl\"");
$output = shell_exec($command);

if (!$output) {
    $output = "[Error: Keine Antwort vom Python Script erhalten.]";
}
?>

<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Transkript anzeigen</title>
    <style>
        body {
            background-color: #121212;
            color: #e0e0e0;
            font-family: 'Segoe UI', sans-serif;
            padding: 20px;
        }
        h1 {
            font-size: 2rem;
            text-align: center;
        }
        pre {
            background-color: #1e1e1e;
            padding: 20px;
            border-radius: 8px;
            white-space: pre-wrap;
            font-size: 1rem;
            overflow-x: auto;
        }
        .video-button {
            display: block;
            margin: 30px auto;
            background-color: #28a745;
```

```
        color: white;
        padding: 15px 30px;
        text-align: center;
        text-decoration: none;
        font-size: 1.2rem;
        border-radius: 8px;
        width: max-content;
    }
</style>
</head>
<body>

<h1> Transkript</h1>
<pre><?= htmlspecialchars($output) ?></pre>

<a class="video-button" href="watch_video_embed.php?video_id=<?= urlencode($videoid) ?>">▶ Jetzt Video ansehen</a>

</body>
</html>
```

watch_video.php

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);

define('BASE_PATH', dirname(__DIR__));
require_once BASE_PATH . '/models/SessionManager.php';

SessionManager::startSession();

if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/login.php");
    exit();
}

$userId = $_SESSION['user_id'] ?? 1;
$pythonPath = "python";
$scriptPath = "C:/xampp/htdocs/LingoLoop/models/youtube.py";
$command = escapeshellcmd("$pythonPath \"$scriptPath\" $userId");
$output = shell_exec($command);

$videoList = json_decode($output, true);
?>
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Wähle ein Video</title>
    <style>
        body {
            background-color: #121212;
            color: #e0e0e0;
            font-family: 'Segoe UI', sans-serif;
            margin: 0;
            padding: 30px 15px;
        }

        .header h1 {
            text-align: center;
            font-size: 2.5rem;
            margin-bottom: 30px;
            text-transform: uppercase;
        }

        .videos {
            display: grid;
            grid-template-columns: 1fr;
            gap: 20px;
        }

        @media (min-width: 768px) {
            .videos {
```

```
        grid-template-columns: repeat(2, 1fr);
    }
}

.card {
    background: #1e1e1e;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.5);
    display: flex;
    flex-direction: column;
    align-items: center;
}

.card img {
    width: 100%;
    border-radius: 8px;
    margin-bottom: 12px;
}

.card h3 {
    margin: 0 0 10px;
    font-size: 1.1rem;
    text-align: center;
}

.card p {
    margin: 0 0 10px;
    color: #ccc;
    font-size: 0.95rem;
    text-align: center;
}

.card-footer {
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 100%;
    margin-top: 10px;
}

.duration {
    font-size: 1.1rem;
    font-weight: bold;
    color: #ffc107;
}

.select-radio {
    transform: scale(1.4);
    cursor: pointer;
}

.actions {
```

```

        margin-top: 40px;
        text-align: center;
    }

    .next-button, .dashboard-button {
        margin: 10px;
        padding: 15px 30px;
        font-size: 1.2rem;
        font-weight: bold;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        transition: 0.3s ease;
    }

    .next-button {
        background-color: #007bff;
        color: white;
    }

    .next-button:disabled {
        background-color: #555;
        cursor: not-allowed;
    }

    .dashboard-button {
        background-color: #6c757d;
        color: white;
        text-decoration: none;
    }

    .next-button:hover:enabled {
        background-color: #0056b3;
    }

    .dashboard-button:hover {
        background-color: #5a6268;
    }
</style>
</head>
<body>

<h1 class="header"> Wähle ein Video</h1>

<form action="select_video.php" method="get" id="videoForm">
    <div class="videos">
        <?php
            if (is_array($videoList)) {
                foreach ($videoList as $index => $video) {
                    [$title, $description, $duration, $url] = $video;
                    parse_str(parse_url($url, PHP_URL_QUERY), $queryParams);
                    $videoid = $queryParams['v'] ?? '';
                    $thumbnailUrl = "https://img.youtube.com/vi/$videoid/hqdefault.jpg";
                }
            }
        </?php>
    </div>
</form>

```



```

        echo "
        <label class='card'>
            <img src='$thumbnailUrl' alt='Thumbnail'>
            <h3>" . htmlspecialchars($title) . "</h3>
            <p>" . htmlspecialchars($description) . "</p>
            <div class='card-footer'>
                <span class='duration'> $duration</span>
                <input type='radio' name='video_id' value='$videoid' class='select-radio' required>
            </div>
        </label>
        ";
    }
} else {
    echo "<p>⚠ Keine Videos gefunden.</p>";
}
?>
</div>

<div class="actions">
    <button type="submit" class="next-button" id="nextBtn" disabled>Weiter →</button>
    <a href="/lingoloop/view/dashboard.php" class="dashboard-button"> Zurück zum Dashboard</a>
</div>
</form>

<script>
const radios = document.querySelectorAll('input[name="video_id"]');
const nextBtn = document.getElementById('nextBtn');

radios.forEach(radio => {
    radio.addEventListener('change', () => {
        nextBtn.disabled = false;
    });
});
</script>

</body>
</html>

```

welcome.php

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);

define('BASE_PATH', dirname(__DIR__));
require_once BASE_PATH . '/models/SessionManager.php';

SessionManager::startSession();

if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/login.php");
    exit();
}

// === USER ID iz sesije ili hardkodirano ===
$userId = $_SESSION['user_id'] ?? 1; // Ako nema u sesiji, koristi 1

// === Putanja do Pythona i skripte ===
$pythonPath = "python";
$scriptPath = "C:/xampp/htdocs/LingoLoop/models/youtube.py";

// === Komanda za izvršenje ===
$command = escapeshellcmd("$pythonPath \"$scriptPath\" $userId");

// === Izvršavanje i output ===
$output = shell_exec($command);

// === HTML Prikaz ===
echo "<!DOCTYPE html>
<html lang='en'>
<head>
    <meta charset='UTF-8'>
    <title>LingoLoop | Top 54 Videos</title>
    <style>
        body {
            background-color: #121212;
            color: #e0e0e0;
            font-family: 'Segoe UI', sans-serif;
            padding: 20px;
        }
        pre {
            white-space: pre-wrap;
            background: #1e1e1e;
            padding: 20px;
            border-radius: 10px;
            overflow-x: auto;
            font-size: 1rem;
        }
        .next-button {
            position: absolute;
            top: 20px;
```

```
    right: 20px;
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 6px;
    font-size: 1rem;
    cursor: pointer;
    text-decoration: none;
    transition: background-color 0.3s ease;
  }
  .next-button:hover {
    background-color: #0056b3;
  }
</style>
</head>
<body>

  <a href='/lingoloop/view/next_video.php' class='next-button'>Next →</a>

  <h1> Top 54 YouTube Videos</h1>
  <pre>" . htmlspecialchars($output) . "</pre>

</body>
</html>";
?>
```