

```
├─ ./
│  .gitattributes
│  DejaVuSans.ttf
│  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
│           └─ remotes/
│              └─ origin/
│                 HEAD
│                 login-register-feature
│                 main
│     └─ objects/
│        └─ 00/
│           8c2718817cb0b4923aa615e7efd56180cab6f9
│        └─ 03/
│           5496e63b56af046bd1286ed29c5ad981808e1d
│        └─ 05/
│           0e0cc407b0a2bed50f089188e78216d6015516
│        └─ 0c/
│           c68ad796a26df76448bee297d9b83e6cb1c13c
│        └─ 0d/
```

5ca5da88199118bf964bb0d8fea0168d884d27  
— 17/  
7854d8fb070915778ae3eada4fbca40af9d984  
— 1c/  
4382a0c2cf21edc14e94a90ddfb25442e16cdc  
— 23/  
d0a34f1ea8d7133b10f2c1a11eaaebf4d21a3b  
— 26/  
9ecd616f3c52b067e691114397f6a47a1aee4d  
— 28/  
2646dcb22cef2f623b14e741f1047bcab939b0  
— 2c/  
91ec2ff8eb8bcee28bf8794c4341e0909dad9c  
— 31/  
440a23beafbc3ecfa5e906dc41a63fd95ee31b  
— 33/  
01408c50e49807fabd69ef0cecae3e4ab5d2b0  
— 39/  
8511bc9cab44e252179209486ce68927658c3b  
— 3c/  
b94bc5bca6344e45a1cc63f2e1cfbb4a6fe0ca  
— 3e/  
53aa29d8790b8bbf90d0abcd611f9e4a91a122  
6187a8fc255d0939ab3e20963d14dc531ed4f2  
— 41/  
70cf6f9ef167d036071b81ffec588c550ffd8c  
— 42/  
cca7de0bb3bec81351e5611e30dc0e5442a4e0  
— 46/  
39a08f179604d605afcaeeae4457569acffc9b  
— 4d/  
3b76edaf07feb8e0138e399dd78a88b2f65321  
daa9873f23907a3092c2246d79bdbb419cb065  
— 4e/  
16863c17a5685167067646195436938419c9be  
— 4f/  
9bbdadff091c8e5240e8103eca70c4076d2b0b  
— 57/  
b4ee1b0cb7c2579ad6071539e094f35da50fef  
— 58/  
c38ffd6a49b1284b979457ea76c53dae83c280  
— 5b/  
14bdae48f2638c9d12fb7597593f4363c0a0fa  
— 5e/  
0e0fc8da6323e9370892e3945037ca1bda9cc0  
— 64/  
60a5ae736675b80f63419d518f83fe6ef374aa  
64e285ee2f2fc61f5c2725166d2a8e572de24b  
— 69/  
bd7a2bff2f0f5e04ec2d08ab98f2a0053a661e  
— 6b/  
22ade825fd85a0f1a82f4341da5b54f5044968  
— 6f/

9509c88bed7080d496fc5e1d87a9315e30549d  
— 74/  
d9690b6d4d6a1c81fb212e88a319922345f733  
— 76/  
045c9331dcf0c817619bbbc908074c09626900  
— 77/  
9d49f6b2a848dc8e2265860ec75cf2c050359d  
— 84/  
cefba8cd023cac1784c835bd62fa3dc4f5351e  
— 85/  
0eddc73e06fa3f619f36f2eb541416e91b0acc  
— 86/  
8a637b051da6c8b101f800ab60bb68ee5ea2db  
— 87/  
132b042e96065ee35426f3f2085e8497269025  
— 8a/  
c81a3c49d062a279d36018134519aed939151c  
— 8b/  
2820b754ab0100db9f9f1d24a585dcf82bea27  
— 8c/  
89f4a0309a312f33371be10898ae413c1dde41  
— 95/  
4e8ef3c7722dca22768af4eb181d3b91333bec  
68d7e82c8b5dd5578fa68d0798110f88f48f46  
— 96/  
8e433211f5c403512b39e7af9ff00f0d03b7b2  
— 97/  
9687755aa5345a63b4acf0b34fe8ded34e5bdf  
— 9e/  
0267e4e8c2f2d4d5ec6f3c809cc39da20751d2  
ee275518aa5b6ea282f0ae616c2480f4da2d20  
— 9f/  
329eb66ddb57a8feaf0f61141450fe79bb85ef  
— a0/  
d7407156d2757d82c5c700b8ce8e41a4c13100  
— a2/  
7f98f6b7cfae2185592a71c0b513934113fcf7  
— a3/  
ded5cbbe298242b5f25470389c877a5a426db8  
— a6/  
7df1f1cc80b6d9c9b3854236986d094fa3657f  
— af/  
54e267b39268ed512a8a72a5b3fc8dc0db0c31  
— b3/  
d786489ece5f5deddfefec23740bfac5207af4  
— b6/  
1b44162a4db1c9c79e5b716d7fd4fc26f506c5  
— b8/  
2d87d49f9c2945e5292c14e261d60a401e8805  
— c0/  
9c9c0036c9b3daeb218f6af919783035f63668  
— c2/  
2f1993c00680e5c3b217233ea93a38d86ea610

- └─ c9/
  - 119437c302d5ea6460ae9aabb18406abe92cf
- └─ cc/
  - 31f7126189c3bbc7a6e5b50567ea9a2e4af042
- └─ cf/
  - 2fa27a0a38dca3a2527745d2d364b339c75aa3
- └─ d5/
  - c017fd905c05398926d7e246b9063f9caac014
  - f59842ca9c27376709c4b881d52e45a64232bd
- └─ d9/
  - f22f08f494e6c7850ac3180b8830f28e13e2d9
- └─ df/
  - e0770424b2a19faf507a501ebfc23be8f54e7b
- └─ e1/
  - 2086927223f320493b805f03bc697253098143
- └─ e3/
  - 356862bfd2c8e8b8a6c40557d32b20ef339f61
- └─ e5/
  - 25e6d7839a48722c6e49a0d536a23095dae4c3
  - 9a4c5f4239675ff85d24af359a933bd1e13927
  - babd58a1b8bb7f77f761f3f08da6465a3c1059
- └─ e7/
  - f08517e9939eb4981f437cf8548391625b0d26
- └─ ef/
  - 417601ddaf9e159d8540a7520b8512d777701e
- └─ f1/
  - 29bb3b1264fc8331504ad0e67609e88e982e11
- └─ f2/
  - 2c9cc73c7e5d34640e1058f8e09b8e315fb059
- └─ f5/
  - 1d5ff2eb7ecc32cfc3bfc4b9501de4c7f5fa77
- └─ f6/
  - ba88f6595730634e54411278dbdeca9a80127a
- └─ f9/
  - db032d3ac050aa543e3aa68f5d303cbcbcb2443
- └─ fa/
  - 73fc3d9f539f0233f4adb62e037af956175e15
- └─ fb/
  - b96afa731457bfdafdb646bdd1a9821ec36e40
- └─ fc/
  - 51eb38017a8b69b1c5af2a09b748fa5e1ffeed
- └─ fe/
  - 397c397cfe428a75b711902844caa2564196f0
- └─ info/
- └─ pack/
- └─ refs/
  - └─ heads/
    - ai-vocabulary-generation
    - login-register-feature
    - main
  - └─ remotes/
    - └─ origin/
- └─ HEAD

```
| | | | login-register-feature
| | | | main
| | | |─ tags/
|─ .htaccess/
|─ app/
|  test.py
|  |─ controllers/
|  |  AuthController.php
|  |  test.py
|  |  |─ __pycache__/
|  |  |  test.cpython-311.pyc
|  |─ models/
|  |  ai_vocabulary_generation.py
|  |  User.php
|  |  |─ __pycache__/
|  |  |  ai_vocabulary_generation.cpython-311.pyc
|  |─ services/
|  |─ views/
|  |  dashboard.php
|  |  |─ auth/
|  |  |  login.php
|  |  |  register.php
|  |  |─ dashboard/
|  |  |  welcome.php
|  |  |─ profile/
|  |  |  setup_profile.php
|  |─ __pycache__/
|  |  test.cpython-311.pyc
|─ config/
|  config.php
|  config.py
|  keys.json
|  __init__.py
|  |─ __pycache__/
|  |  config.cpython-311.pyc
|  |  __init__.cpython-311.pyc
|─ core/
|  Database.php
|  Database.py
|  SessionManager.php
|  __init__.py
|  |─ __pycache__/
|  |  Database.cpython-311.pyc
|  |  __init__.cpython-311.pyc
|─ Diagrams/
|  |─ Activity Diagram/
|  |  Login_Register Activity Diagram v0.0.png
|  |─ Structure/
|  |  LingoLoop_Project_Structure.pdf
|  |─ Use Case/
|  |  Use Case v0.0.pdf
|─ helpers/
|─ public/
```

```
| | .htaccess
| | index.php
| | logout.php
| | SaveProfileController.php
| | test.php
| └─ routes/
| └─ storage/
```

# 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()

```



## test.py

```
from core.Database import Database
connection = Database.get_instance()

if connection.is_connected():
    print(" Konekcija sa bazom je uspešna.")
else:
    print(" Konekcija nije uspela.")
```

# AuthController.php

```
<?php

require_once __DIR__ . '/../models/User.php';
require_once __DIR__ . '/../core/SessionManager.php';
require_once __DIR__ . '/../core/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']);

            // Proveri da li korisnik već ima profil
            if (!$this->userModel->hasProfile($user['id'])) {
                header("Location: /lingoloop/public/index.php?action=setup_profile");
                exit();
            }

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

        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/public/index.php?action=setup_profile");
            exit();
        }
    }
}
```

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

## test.py

```
from core.Database import Database
connection = Database.get_instance()

if connection.is_connected():
    print(" Konekcija sa bazom je uspešna.")
else:
    print(" Konekcija nije uspela.")
```

## ai\_vocabulary\_generation.py

```
import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..', '..')))
from core.Database import *
from groq import Groq
import json
import ast

class AI_VOCABULARY_GENERATION:
    def __init__(self):
        self.__connection = Database.get_instance()
        self.__cursor = self.__connection.cursor()
        with open('config/keys.json', '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:
            output += " Red:\n"
            for col_name, value in zip(column_names, red):
                output += f"{col_name}: {value}\n"
            output += "-----\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"
                        "Beziehe dich auf seine Interessen, Ziele und seinen Lernstil, um relevante Ausdrücke zu wählen.\n\n"
                    )
                }
            ]
        )
```

```

        """Gib für jeden Begriff Folgendes an:**\n"
        "1. Der englische Begriff\n"
        "2. Die passende deutsche Übersetzung *(mit Artikel bei Substantiven)**\n"
        "3. Einen kurzen Beispielsatz auf Englisch\n\n"
        "Format:\n"
        "[\n"
        " ('term1', 'Übersetzung1', 'Beispielsatz1'),\n"
        " ('term2', 'Übersetzung2', 'Beispielsatz2'),\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:
    print("Parsing error:", e)
    print("Raw response:", response_text)
    return []

```

```

x = AI_VOCABULARY_GENERATION()
text = x.getting_data_from_ab(1)

```

```

print(x.create_vocab(text))

```

# 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;
}

}

```



# dashboard.php

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

// Check if user is logged in
SessionManager::start();
if (!SessionManager::get('user_id')) {
    header("Location: /index.php");
    exit();
}

$username = SessionManager::get('username');
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>LingoLoop | Dashboard</title>
</head>
<body>
    <h1>Welcome to LingoLoop, <?= htmlspecialchars($username) ?>!</h1>

    <p>This is your user dashboard.</p>

    <p><a href="/logout.php">Logout</a></p>
</body>
</html>
```

# login.php

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

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

<!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="/lingoloop/public/index.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/public/?action=register">Register here</a></p>
  </div>
</body>
</html>
```

# register.php

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

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

<!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/public/index.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/public/?action=login">Login here</a></p>
</div>
</body>
</html>
```

# welcome.php

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

SessionManager::startSession();

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

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

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome | LingoLoop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-black text-white flex items-center justify-center h-screen">

    <div class="bg-gray-900 p-10 rounded-xl shadow-lg text-center max-w-md w-full">
        <h1 class="text-3xl font-bold mb-4">Welcome, <?php echo htmlspecialchars($username); ?>! </h1>
        <p class="text-gray-300 mb-6">You are successfully logged in to <span class="font-semibold
text-white">LingoLoop</span>. </p>

        <a href="/lingoloop/public/logout.php" class="bg-red-600 hover:bg-red-700 transition px-6 py-2 rounded text-white
font-semibold">
            Logout
        </a>
    </div>

</body>
</html>
```



# setup\_profile.php

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

SessionManager::startSession();

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

$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/public/SaveProfileController.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 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>
```

```

</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>
```

# 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`

# 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):
        print(DB_CONFIG)

        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
```

# 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;  
}
```

```
}
```

`__init__.py`

# index.php

```
<?php
define('BASE_PATH', dirname(__DIR__));
require_once BASE_PATH . '/app/controllers/AuthController.php';
require_once BASE_PATH . '/core/SessionManager.php';

$auth = new AuthController();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $action = $_POST['action'] ?? null;

    if ($action === 'login') {
        $username = trim($_POST['username']);
        $password = trim($_POST['password']);
        $error = $auth->login($username, $password);

        if ($error) {
            include BASE_PATH . '/app/views/auth/login.php';
        }
    }

    if ($action === 'register') {
        $username = trim($_POST['username']);
        $email = trim($_POST['email']);
        $password = trim($_POST['password']);
        $error = $auth->register($username, $email, $password);

        if ($error) {
            include BASE_PATH . '/app/views/auth/register.php';
        }
    }
} else {
    $action = $_GET['action'] ?? null;

    if ($action === 'register') {
        include BASE_PATH . '/app/views/auth/register.php';
    } elseif ($action === 'welcome') {
        include BASE_PATH . '/app/views/dashboard/welcome.php';
    } elseif ($action === 'login') {
        include BASE_PATH . '/app/views/auth/login.php';
    } else {
        include BASE_PATH . '/app/views/profile/setup_profile.php';
    }
}
```

# logout.php

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

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

```
header("Location: /lingoloop/public/index.php");  
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;
$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'] ?? '';

$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()) {
        header("Location: /lingoloop/public/index.php?action=welcome");
        exit();
    } else {
        echo "Error saving profile: " . $stmt->error;
    }
}
```

test.php

```
<?php  
echo "Apache radi!";  
?>
```