```
├── ./
│   .gitattributes
│   .gitignore
│   DejaVuSans.ttf
│   final_output.pdf
│   greske.txt
│   project_to_pdf.py
│   ├── config/
│   │   config.php
│   │   config.py
│   │   keys.json
│   │   __init__.py
│   ├── controller/
│   │   AuthController.php
│   │   delete_word.php
│   │   LogoutController.php
│   │   SaveProfileController.php
│   │   SaveVocabController.php
│   │   save_word.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_translation.py
│   │   ai_vocabulary_generation.py
│   │   Database.php
│   │   Database.py
│   │   SaveProfile.php
│   │   SaveVocab.php
│   │   SessionManager.php
│   │   User.php
│   │   VocabularyManager.php
│   │   youtube.py
│   ├── view/
│   │   add_word.php
│   │   dashboard.php
│   │   delete_word.php
│   │   index.php
│   │   learn_words.php
│   │   login.php
│   │   register.php
│   │   revise_words.php
│   │   select_video.php
│   │   select_vocab.php
│   │   setup_profile.php
│   │   show_transcript.php
│   │   translate.php
│   │   translations_list.php
│   │   vocabulary_dashboard.php
│   │   watch_video.php
│   │   watch_video_embed.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"
EXCLUDE_DIRS = {"__pycache__", ".git", "venv", "env", ".idea", ".vscode"}

# Generate a visual folder structure as a string
def generate_structure_text(root_dir):
    structure = ""
    for dirpath, dirnames, filenames in os.walk(root_dir):
        # Skip excluded directories
        if any(excluded in dirpath.split(os.sep) for excluded in EXCLUDE_DIRS):
            continue

        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 (excluding unwanted dirs)
def get_all_code_files(root, extensions=(".py", ".php", ".html")):
    collected = []
    for dirpath, dirnames, filenames in os.walk(root):
        if any(excluded in dirpath.split(os.sep) for excluded in EXCLUDE_DIRS):
            continue
        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
<?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']);

            //  Ako korisnik nema red u user_profiles → pokreni LLM
            if (!$this->userModel->hasProfileRow($user['id'])) {
                $scriptPath = BASE_PATH . "/models/ai_vocabulary_generation.py";
                $command = "python \"$scriptPath\" {$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/select_vocab.php");
                    exit();
                } else {
                    echo "Error decoding vocabulary list.";
                    exit();
                }
            }

            //  Ako profil postoji → idi na dashboard
            header("Location: /lingoloop/view/dashboard.php");
            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) {
            $user = $this->userModel->findByUsername($username);
            SessionManager::start($user['id'], $user['username']);
            header("Location: /lingoloop/view/setup_profile.php");
            exit();
        }

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

delete_word.php

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

SessionManager::startSession();

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

$db = Database::getInstance();
$userId = $_SESSION['user_id'] ?? null;
$vocabManager = new VocabularyManager($db);

// Brisanje reči
if (isset($_GET['delete'])) {
    $wordId = intval($_GET['delete']);
    $vocabManager->deleteWord($wordId);
    header("Location: delete_word.php");
    exit();
}

$allWords = $vocabManager->getAllWords($userId);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Delete Words</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            background-color: #000;
            color: white;
            font-family: Helvetica, Arial, sans-serif;
            margin: 0;
            padding: 40px 20px;
        }

        .container {
            max-width: 700px;
            margin: 0 auto;
            text-align: center;
        }

        h1 {
            font-size: 2.5rem;
    margin-bottom: 30px;
}

.button {
    padding: 14px 26px;
    font-size: 1.1rem;
    border: none;
    border-radius: 8px;
    margin: 10px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    text-decoration: none;
    display: inline-block;
}

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

.search-btn {
    background-color: #28a745;
    color: white;
}

.button:hover {
    opacity: 0.85;
}

.search-box {
    width: 100%;
    max-width: 400px;
    padding: 10px;
    margin: 30px auto;
    font-size: 1.1rem;
    border-radius: 6px;
    border: none;
}

.word-card {
    background-color: #111;
    padding: 16px;
    margin-bottom: 12px;
    border-radius: 6px;
    text-align: center;
}

.word-card span {
    font-size: 1.1rem;
    display: block;
    margin-bottom: 10px;
}
```

```
        .delete-btn {
            background-color: #dc3545;
            color: white;
            padding: 8px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    </style>
    <script>
        function filterWords() {
            let input = document.getElementById('searchInput').value.toLowerCase();
            let cards = document.getElementsByClassName('word-card');

            for (let card of cards) {
                let text = card.innerText.toLowerCase();
                card.style.display = text.includes(input) ? 'block' : 'none';
            }
        }
    </script>
</head>
<body>

<div class="container">
    <h1> Delete Words</h1>

    <a href="vocabulary_dashboard.php" class="button menu-btn"> Words Menu</a>

    <input type="text" id="searchInput" onkeyup="filterWords()" class="search-box" placeholder=" Search words...">

    <?php if (empty($allWords)): ?>
        <p>No words found.</p>
    <?php else: ?>
        <?php foreach ($allWords as $word): ?>
            <div class="word-card">
                <span><?= htmlspecialchars($word['term']) ?> – <?= htmlspecialchars($word['translation']) ?></span>
                <form method="GET" onsubmit="return confirm('Are you sure you want to delete this word?');">
                    <input type="hidden" name="delete" value="<?= $word['id'] ?>">
                    <button type="submit" class="delete-btn">Delete</button>
                </form>
            </div>
        <?php endforeach; ?>
    <?php endif; ?>
</div>

</body>
</html>
```

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
<?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(
    "isssssssssss",
    $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
<?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();
```

save_word.php

```
<?php
// Test verzija za provjeru je li pozvan save_word.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();
}
echo json_encode(['success' => true, 'message' => 'save_word.php je uspješno pozvan']);
require_once __DIR__ . '/../models/Database.php';
require_once __DIR__ . '/../models/SaveVocab.php';



$index = isset($_POST['index']) ? intval($_POST['index']) : -1;
$userId = $_SESSION['user_id'] ?? null;
$vocabList = $_SESSION['translations'] ?? [];
$term = $vocabList[$index]['term'];
$translation = $vocabList[$index]['translation'];
$db = Database::getInstance();
$model = new UserVocabulary($db);
$model->saveSelectedWord($userId, $term, $translation);
// Ukloni spremljenu riječ iz sesije
unset($_SESSION['translations'][$index]);

// Reindeksiraj niz da ne ostanu rupe u indeksima
$_SESSION['translations'] = array_values($_SESSION['translations']);
```

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(z)
    print(k)




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

ai_translation.py

```
# ai_translation.py
import sys
import json
from groq import Groq
import os
sys.path.append("C:/xampp/htdocs/LingoLoop")
sys.stdout.reconfigure(encoding='utf-8')

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_key = data["Groq"]
client = Groq(api_key=groq_key)

def translate_text(text):
    prompt = f"""
Translate the following English word or phrase to German.

Only return the result as a tuple in Python format: (term, translation)
Do **not** include any explanation, commentary, formatting or extra text.

Text:
{text}
"""

    completion = client.chat.completions.create(
        model="llama-3.3-70b-versatile",
        messages=[
            {"role": "system", "content": "You are a professional translator fluent in English and German."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.5,
        max_tokens=200
    )

    translated = completion.choices[0].message.content.strip()
    return translated

if __name__ == "__main__":
    input_text = sys.argv[1]
    result = translate_text(input_text)

    if result.startswith("(") and result.endswith(")"):
        print(result)
    else:
        print(f"(\"{input_text}\", \"[translation error]\")")
```

ai_vocabulary_generation.py

```python
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"
                    "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'),\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
<?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
<?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
<?php

class UserVocabulary
{
    private \mysqli $db;

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

    /**
     * Sprema više riječi na temelju indeksa iz liste.
     */
    public function saveSelectedWords(int $userId, array $vocabList, array $selectedIndexes): void
    {
        $stmt = $this->db->prepare("INSERT INTO user_vocabulary
            (user_id, term, translation, date_added, points, next_review_date)
            VALUES (?, ?, ?, NOW(), 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();
    }

    /**
     * Sprema jednu riječ direktno.
     */
    public function saveSelectedWord(int $userId, string $term, string $translation): void
    {
        $stmt = $this->db->prepare("
            INSERT INTO user_vocabulary
                (user_id, term, translation, date_added, points, next_review_date)
            VALUES (?, ?, ?, NOW(), 0, NOW())"
        );

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

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

        if (!$stmt->execute()) {
            throw new Exception("Greška pri izvršavanju upita: " . $stmt->error);
        }

        $stmt->close();
    }

    /**
     * Provjerava ima li korisnik već spremljene riječi.
     */
    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
<?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
<?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;
    }

    public function hasProfileRow(int $userId): bool {
        $stmt = $this->db->prepare("SELECT 1 FROM user_profiles WHERE user_id = ? LIMIT 1");
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $stmt->store_result();
        return $stmt->num_rows > 0;
    }


}
```

VocabularyManager.php

```
<?php

class VocabularyManager
{
    private \mysqli $db;

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

    public function translation_to($term) {
        $escapedInput = escapeshellarg($term);
        $pythonScript = __DIR__ . "/ai_translation.py";
        $command = "python " . escapeshellarg($pythonScript) . " $escapedInput 2>&1";

        $output = shell_exec($command);
        $translation = json_decode($output, true);

        if ($translation === null) {
            error_log(" Translation failed. Raw output: $output");
        }

        return $translation;
    }

    public function addWordToProfile($userId, $word, $translation)
    {
        $stmt = $this->db->prepare("INSERT INTO user_vocabulary
            (user_id, term, translation, date_added,  points, next_review_date, review_days)
            VALUES (?, ?, ?, NOW(),  0, NOW(),0)");

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

    public function getWordsAddedToday($userId)
    {
        $stmt = $this->db->prepare("SELECT COUNT(*) FROM user_vocabulary WHERE user_id = ? AND date_added = CURDATE()");
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $stmt->bind_result($count);
        $stmt->fetch();
        $stmt->close();
        return $count;
    }

    public function getWordsToLearn($userId)
    {
        $stmt = $this->db->prepare("SELECT * FROM user_vocabulary WHERE user_id = ? AND to_learn = 1");
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $result = $stmt->get_result();
        return $result->fetch_all(MYSQLI_ASSOC);
    }

    public function countWordsToLearn($userId)
    {
    $stmt = $this->db->prepare("SELECT COUNT(*) FROM user_vocabulary WHERE user_id = ? AND to_learn = 1");
    $stmt->bind_param("i", $userId);
    $stmt->execute();
    $stmt->bind_result($count);
    $stmt->fetch();
    return $count;
    }

    public function setWordLearned($wordId)
    {
        $stmt = $this->db->prepare("UPDATE user_vocabulary SET  to_learn = 0,review_days = review_days + 1, next_review_date =
CURDATE() + INTERVAL review_days DAY WHERE id = ?");
        $stmt->bind_param("i", $wordId);
        $stmt->execute();
        $stmt->close();
    }

    public function reducePoints($wordId)
    {
        $stmt = $this->db->prepare("UPDATE user_vocabulary SET points = points - 10 WHERE id = ?");
        $stmt->bind_param("i", $wordId);
        $stmt->execute();
        $stmt->close();
    }

    public function getWordsToRevise($userId)
    {
        $stmt = $this->db->prepare("SELECT * FROM user_vocabulary WHERE user_id = ? AND next_review_date <= CURDATE()");
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $result = $stmt->get_result();
        return $result->fetch_all(MYSQLI_ASSOC);
    }

    public function countWordsToRepeat($userId)
    {
        $stmt = $this->db->prepare("SELECT COUNT(*) FROM user_vocabulary WHERE user_id = ? AND to_learn = 0 AND
next_review_date <= CURDATE() ");
    $stmt->bind_param("i", $userId);
    $stmt->execute();
    $stmt->bind_result($count);
    $stmt->fetch();
    return $count;
    }
    public function updateRevision($wordId, $result)
    {
        if ($result === "Yes") {
                $stmt = $this->db->prepare("UPDATE user_vocabulary SET points = points + 10, review_days = review_days + 1,
next_review_date = CURDATE() + INTERVAL review_days DAY WHERE id = ?");
        } else {
            $stmt = $this->db->prepare("UPDATE user_vocabulary SET points = points - 15, review_days = 0 WHERE id = ?");
        }
        $stmt->bind_param("i", $wordId);
        $stmt->execute();
        $stmt->close();
    }

    public function deleteWord($wordId)
    {
        $stmt = $this->db->prepare("DELETE FROM user_vocabulary WHERE id = ?");
        $stmt->bind_param("i", $wordId);
        $stmt->execute();
        $stmt->close();
    }

    public function getAllWords($userId)
    {
        $stmt = $this->db->prepare("SELECT * FROM user_vocabulary WHERE user_id = ? ORDER BY date_added DESC");
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $result = $stmt->get_result();
        return $result->fetch_all(MYSQLI_ASSOC);
    }

    public function getLeastPointWords($userId)
    {
        $stmt = $this->db->prepare("SELECT term FROM user_vocabulary WHERE user_id = ? AND to_learn = 0 ORDER BY points ASC
LIMIT 20");
        $stmt->bind_param("i", $userId);
        $stmt->execute();
        $result = $stmt->get_result();
        return $result->fetch_all(MYSQLI_ASSOC);
    }
}
```

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')
sys.path.append("C:/xampp/htdocs/LingoLoop")
from models.Database import *



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)
        self.__connection = Database.get_instance()
        self.__cursor = self.__connection.cursor()




    def get_watched_video_ids(self, user_id):
        try:
            self.__cursor.execute("SELECT video_id FROM watched_videos WHERE user_id = %s", (user_id,))
            result = self.__cursor.fetchall()
            return {row[0] for row in result}
        except Exception as e:
            print(f"[Error fetching watched videos] {e}")
            return set()



    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, user_id, max_results=50, min_views=50000, return_limit=3):
        youtube = build("youtube", "v3", developerKey=self.__API_key)
```

```
# Pretraga YouTube-a
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"]]

# Dohvat dodatnih podataka o videima
videos_response = youtube.videos().list(
    part="snippet,contentDetails,statistics",
    id=",".join(video_ids)
).execute()

# Dohvati već gledane video ID-eve
watched_ids = self.get_watched_video_ids(user_id)
filtered_videos = []

# Obrada svakog videa
for item in videos_response["items"]:
    video_id = item["id"]

    # Preskoči ako je već gledan
    if video_id in watched_ids:
        continue

    try:
        # Filtracija po trajanju i gledanosti (bez provjere transkripta)
        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={video_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, user_id=user_id)
            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
            user_id = int(param)
            result = yt.fetch_top_video_summaries(user_id=user_id, max_videos=10, top_n=4)
            print(json.dumps(result, ensure_ascii=False, indent=2))

    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.]")
```

add_word.php

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

SessionManager::startSession();

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

$db = Database::getInstance();
$userId = $_SESSION['user_id'] ?? null;
$vocabManager = new VocabularyManager($db);

// Dodavanje reči
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $word = trim($_POST['term']);
    if ($word) {
        $translation = $vocabManager->translation_to($word);
        if ($translation) {
            $vocabManager->addWordToProfile($userId, $word, $translation);
            header("Location: add_word.php");
            exit();
        } else {
            echo "<p style='color:red;'> Translation failed. Check your internet connection or AI service.</p>";
        }
    }
}

// Prikaz današnjih reči
$todayWords = array_filter(
    $vocabManager->getAllWords($userId),
    fn($word) => substr($word['date_added'], 0, 10) === date('Y-m-d')
);

// Brisanje reči
if (isset($_GET['delete'])) {
    $wordId = intval($_GET['delete']);
    $vocabManager->deleteWord($wordId);
    header("Location: add_word.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
<title>Add Word (Auto Translation)</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
    body {
        background-color: #000;
        color: #fff;
        font-family: Arial, sans-serif;
        padding: 20px;
    }

    h2 {
        text-align: center;
        margin-bottom: 20px;
    }

    form {
        max-width: 600px;
        margin: 0 auto 30px auto;
    }

    input[type="text"] {
        width: 100%;
        padding: 12px;
        font-size: 1rem;
        margin-bottom: 10px;
        border-radius: 6px;
        border: none;
    }

    button[type="submit"], .menu-btn {
        padding: 12px 20px;
        font-size: 1rem;
        border: none;
        border-radius: 6px;
        background-color: #28a745;
        color: white;
        cursor: pointer;
        margin-top: 10px;
        display: inline-block;
    }

    .menu-btn {
        background-color: #007bff;
        text-decoration: none;
    }

    .scroll-container {
        max-height: 400px;
        overflow-y: auto;
        margin-top: 30px;
        border-top: 1px solid #333;
        padding-top: 20px;
    }
```

```
        .word-button {
            background-color: #111;
            padding: 16px;
            margin-bottom: 12px;
            border-radius: 6px;
            text-align: center;
            display: flex;
            flex-direction: column;
            align-items: center;
        }

        .word-content span {
            font-size: 1.1rem;
            margin-bottom: 10px;
        }

        .delete-btn {
            background-color: #28a745;
            color: white;
            padding: 8px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    </style>
</head>
<body>

<h2> Add Word (Auto-Translate)</h2>

<form method="POST">
    <input type="text" name="term" placeholder="Enter word" required>
    <button type="submit">Submit</button>
    <a href="vocabulary_dashboard.php" class="menu-btn"> Menu</a>
</form>

<div class="scroll-container">
    <h3> Words Added Today</h3>

    <?php if (empty($todayWords)): ?>
        <p>No words added today.</p>
    <?php else: ?>
        <?php foreach ($todayWords as $word): ?>
            <div class="word-button">
                <div class="word-content">
                    <span><?= htmlspecialchars($word['term']) ?> – <?= htmlspecialchars($word['translation']) ?></span>
                </div>
                <form method="GET" onsubmit="return confirm('Delete this word?');">
                    <input type="hidden" name="delete" value="<?= $word['id'] ?>">
                    <button type="submit" class="delete-btn">Delete</button>
                </form>
            </div>
        <?php endforeach; ?>
    <?php endif; ?>
</div>

</body>
</html>
```

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/vocabulary_dashboard.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>
```

delete_word.php

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

SessionManager::startSession();

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

$db = Database::getInstance();
$userId = $_SESSION['user_id'] ?? null;
$vocabManager = new VocabularyManager($db);

// Brisanje reči
if (isset($_GET['delete'])) {
    $wordId = intval($_GET['delete']);
    $vocabManager->deleteWord($wordId);
    header("Location: delete_word.php");
    exit();
}

$allWords = $vocabManager->getAllWords($userId);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Delete Words</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            background-color: #000;
            color: white;
            font-family: Helvetica, Arial, sans-serif;
            margin: 0;
            padding: 40px 20px;
        }

        .container {
            max-width: 700px;
            margin: 0 auto;
            text-align: center;
        }

        h1 {
            font-size: 2.5rem;
```

```css
    margin-bottom: 30px;
}

.button {
    padding: 14px 26px;
    font-size: 1.1rem;
    border: none;
    border-radius: 8px;
    margin: 10px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    text-decoration: none;
    display: inline-block;
}

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

.search-btn {
    background-color: #28a745;
    color: white;
}

.button:hover {
    opacity: 0.85;
}

.search-box {
    width: 100%;
    max-width: 400px;
    padding: 10px;
    margin: 30px auto;
    font-size: 1.1rem;
    border-radius: 6px;
    border: none;
}

.word-card {
    background-color: #111;
    padding: 16px;
    margin-bottom: 12px;
    border-radius: 6px;
    text-align: center;
}

.word-card span {
    font-size: 1.1rem;
    display: block;
    margin-bottom: 10px;
}
```

```
        .delete-btn {
            background-color: #dc3545;
            color: white;
            padding: 8px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    </style>
    <script>
        function filterWords() {
            let input = document.getElementById('searchInput').value.toLowerCase();
            let cards = document.getElementsByClassName('word-card');

            for (let card of cards) {
                let text = card.innerText.toLowerCase();
                card.style.display = text.includes(input) ? 'block' : 'none';
            }
        }
    </script>
</head>
<body>

<div class="container">
    <h1> Delete Words</h1>

    <a href="vocabulary_dashboard.php" class="button menu-btn"> Words Menu</a>

    <input type="text" id="searchInput" onkeyup="filterWords()" class="search-box" placeholder=" Search words...">

    <?php if (empty($allWords)): ?>
        <p>No words found.</p>
    <?php else: ?>
        <?php foreach ($allWords as $word): ?>
            <div class="word-card">
                <span><?= htmlspecialchars($word['term']) ?> – <?= htmlspecialchars($word['translation']) ?></span>
                <form method="GET" onsubmit="return confirm('Are you sure you want to delete this word?');">
                    <input type="hidden" name="delete" value="<?= $word['id'] ?>">
                    <button type="submit" class="delete-btn">Delete</button>
                </form>
            </div>
        <?php endforeach; ?>
    <?php endif; ?>
</div>

</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>
```

learn_words.php

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

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

// Reset session data
if (isset($_GET['reset'])) {
                    unset($_SESSION['learn_data'],    $_SESSION['rot_count'],    $_SESSION['total'],    $_SESSION['answer_revealed'],
$_SESSION['user_input']);
    header("Location: vocabulary_dashboard.php");
    exit();
}

$db = Database::getInstance();
$userId = $_SESSION['user_id'] ?? null;
$vocabManager = new VocabularyManager($db);

// Load words if session not set
if (!isset($_SESSION['learn_data'])) {
    $_SESSION['learn_data'] = array_slice($vocabManager->getWordsToLearn($userId), 0, 10);
    $_SESSION['rot_count'] = 0;
    $_SESSION['total'] = count($_SESSION['learn_data']);
}

// Exit if no words
if (empty($_SESSION['learn_data'])) {
                    unset($_SESSION['learn_data'],    $_SESSION['rot_count'],    $_SESSION['total'],    $_SESSION['answer_revealed'],
$_SESSION['user_input']);
    header("Location: vocabulary_dashboard.php");
    exit();
}

// Handle actions
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action'])) {
    $action = $_POST['action'];
    $current = $_SESSION['learn_data'][0];

    if ($action === 'reveal') {
        $_SESSION['answer_revealed'] = true;
        $_SESSION['user_input'] = $_POST['user_answer'] ?? '';
    }

    if ($action === 'yes') {
        $vocabManager->setWordLearned($current['id']);
        array_shift($_SESSION['learn_data']);
        $_SESSION['rot_count']++;
        unset($_SESSION['answer_revealed'], $_SESSION['user_input']);
    }

    if ($action === 'no') {
        $vocabManager->reducePoints($current['id']);
        array_push($_SESSION['learn_data'], $current);
        array_shift($_SESSION['learn_data']);
        $_SESSION['rot_count']++;
        unset($_SESSION['answer_revealed'], $_SESSION['user_input']);
    }

    // Check if all done
    if (empty($_SESSION['learn_data'])) {
                            unset($_SESSION['learn_data'],    $_SESSION['rot_count'],    $_SESSION['total'],    $_SESSION['answer_revealed'],
$_SESSION['user_input']);
        header("Location: vocabulary_dashboard.php");
        exit();
    }
}

// Get current word and direction AFTER handling actions
$current = $_SESSION['learn_data'][0];
$direction = $_SESSION['rot_count'] % 3 < 2 ? 'DE_EN' : 'EN_DE';
$question = $direction === 'DE_EN' ? $current['translation'] : $current['term'];
$correctAnswer = $direction === 'DE_EN' ? $current['term'] : $current['translation'];

$total = $_SESSION['total'];
$done = $total - count($_SESSION['learn_data']);
$progress = $total > 0 ? round(($done / $total) * 100) : 0;
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Learn Words</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body { background-color: #000; color: white; font-family: Helvetica, Arial, sans-serif; padding: 40px; text-align: center; }
        h2 { font-size: 2rem; margin-bottom: 20px; }
        .word-box { background-color: #111; padding: 30px; font-size: 1.8rem; border-radius: 10px; margin-bottom: 20px; }
        .input-box { padding: 12px; font-size: 1.1rem; width: 60%; border-radius: 6px; border: none; margin-bottom: 20px; }
        .answer { font-size: 1.3rem; margin: 20px auto; color: #ffc107; }
              .progress-container  {  width:  100%;  background-color:  #333;  border-radius:  8px;  overflow:  hidden;  margin:  30px  auto;
max-width: 500px; }
        .progress-bar { height: 20px; background-color: #28a745; width: <?= $progress ?>%; transition: width 0.5s ease-in-out; }
        .progress-label { margin-bottom: 8px; font-size: 1rem; color: #ccc; }
        button { padding: 12px 20px; margin: 5px; border: none; border-radius: 6px; font-size: 1rem; cursor: pointer; }
        .reveal-btn { background-color: #ffc107; color: #000; }
        .yes-btn { background-color: #28a745; color: white; }
        .no-btn { background-color: #dc3545; color: white; }
        .menu-btn { background-color: #007bff; color: white; text-decoration: none; padding: 10px 20px; border-radius: 6px; display: inline-block; margin-top: 20px; }
    </style>
</head>
<body>

<h2> <?= $direction === 'DE_EN' ? "German ➜ English" : "English ➜ German" ?></h2>

<div class="progress-label">Progress: <?= $done ?> / <?= $total ?> learned</div>
<div class="progress-container"><div class="progress-bar"></div></div>

<div class="word-box"><?= htmlspecialchars($question) ?></div>

<form method="POST">
    <?php if (empty($_SESSION['answer_revealed'])): ?>
        <input type="text" name="user_answer" class="input-box" placeholder="Type your translation..." autofocus required>
        <br>
        <button name="action" value="reveal" class="reveal-btn">Reveal</button>
    <?php else: ?>
        <div class="answer">
            Your answer: <strong><?= htmlspecialchars($_SESSION['user_input']) ?></strong><br>
            Correct answer: <strong><?= htmlspecialchars($correctAnswer) ?></strong>
        </div>
        <button name="action" value="yes" class="yes-btn">I Knew It</button>
        <button name="action" value="no" class="no-btn">I Didn't Know</button>
    <?php endif; ?>
</form>

<a href="learn_words.php?reset=1" class="menu-btn"> Back to Menu</a>

</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>
```

revise_words.php

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

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

// Reset session data
if (isset($_GET['reset'])) {
    unset($_SESSION['revise_data'], $_SESSION['rot_count'], $_SESSION['total'], $_SESSION['answer_revealed'],
$_SESSION['user_input']);
    header("Location: vocabulary_dashboard.php");
    exit();
}

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

// Load words to revise
if (!isset($_SESSION['revise_data'])) {
    $_SESSION['revise_data'] = is_array($wordsToRevise) ? array_slice($wordsToRevise, 0, 10) : [];
    $_SESSION['rot_count'] = 0;
    $_SESSION['total'] = count($_SESSION['revise_data']);
}

// Exit if no words
if (empty($_SESSION['revise_data'])) {
    unset($_SESSION['revise_data'], $_SESSION['rot_count'], $_SESSION['total'], $_SESSION['answer_revealed'],
$_SESSION['user_input']);
    header("Location: vocabulary_dashboard.php");
    exit();
}

// Handle user actions
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action'])) {
    $action = $_POST['action'];
    $current = $_SESSION['revise_data'][0];

    if ($action === 'reveal') {
        $_SESSION['answer_revealed'] = true;
        $_SESSION['user_input'] = $_POST['user_answer'] ?? '';
    }

    if ($action === 'yes' || $action === 'no') {
        $vocabManager->updateRevision($current['id'], ucfirst($action));
        array_shift($_SESSION['revise_data']);
        $_SESSION['rot_count']++;
        unset($_SESSION['answer_revealed'], $_SESSION['user_input']);
    }

    if (empty($_SESSION['revise_data'])) {
                              unset($_SESSION['revise_data'],    $_SESSION['rot_count'],    $_SESSION['total'],    $_SESSION['answer_revealed'],
$_SESSION['user_input']);
        header("Location: vocabulary_dashboard.php");
        exit();
    }
}

// Determine direction and current word (after POST)
$current = $_SESSION['revise_data'][0];
$direction = $_SESSION['rot_count'] % 3 < 2 ? 'DE_EN' : 'EN_DE';
$question = $direction === 'DE_EN' ? $current['translation'] : $current['term'];
$correctAnswer = $direction === 'DE_EN' ? $current['term'] : $current['translation'];

$total = $_SESSION['total'];
$done = $total - count($_SESSION['revise_data']);
$progress = $total > 0 ? round(($done / $total) * 100) : 0;
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Revise Words</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body { background-color: #000; color: white; font-family: Helvetica, Arial, sans-serif; padding: 40px; text-align: center; }
        h2 { font-size: 2rem; margin-bottom: 20px; }
        .word-box { background-color: #111; padding: 30px; font-size: 1.8rem; border-radius: 10px; margin-bottom: 20px; }
        .input-box { padding: 12px; font-size: 1.1rem; width: 60%; border-radius: 6px; border: none; margin-bottom: 20px; }
        .answer { font-size: 1.3rem; margin: 20px auto; color: #ffc107; }
              .progress-container  {  width:  100%;  background-color:  #333;  border-radius:  8px;  overflow:  hidden;  margin:  30px  auto;
max-width: 500px; }
        .progress-bar { height: 20px; background-color: #28a745; width: <?= $progress ?>%; transition: width 0.5s ease-in-out; }
        .progress-label { margin-bottom: 8px; font-size: 1rem; color: #ccc; }
        button { padding: 12px 20px; margin: 5px; border: none; border-radius: 6px; font-size: 1rem; cursor: pointer; }
        .reveal-btn { background-color: #ffc107; color: #000; }
        .yes-btn { background-color: #28a745; color: white; }
        .no-btn { background-color: #dc3545; color: white; }
         .menu-btn { background-color: #007bff; color: white; text-decoration: none; padding: 10px 20px; border-radius: 6px; display:
inline-block; margin-top: 20px; }
    </style>
</head>
<body>

<h2> <?= $direction === 'DE_EN' ? "Review: German ➜ English" : "Review: English ➜ German" ?></h2>

<div class="progress-label">Progress: <?= $done ?> / <?= $total ?> revised</div>
```

```
<div class="progress-container"><div class="progress-bar"></div></div>

<div class="word-box"><?= htmlspecialchars($question) ?></div>

<form method="POST">
    <?php if (empty($_SESSION['answer_revealed'])): ?>
        <input type="text" name="user_answer" class="input-box" placeholder="Type your translation..." autofocus required>
        <br>
        <button name="action" value="reveal" class="reveal-btn">Reveal</button>
    <?php else: ?>
        <div class="answer">
            Your answer: <strong><?= htmlspecialchars($_SESSION['user_input']) ?></strong><br>
            Correct answer: <strong><?= htmlspecialchars($correctAnswer) ?></strong>
        </div>
        <button name="action" value="yes" class="yes-btn">I Knew It</button>
        <button name="action" value="no" class="no-btn">I Didn't Know</button>
    <?php endif; ?>
</form>

<a href="revise_words.php?reset=1" class="menu-btn"> Back to Menu</a>

</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;
            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($videoId) ?>"> Transkript anzeigen</a>
        <a class="button" href="/lingoloop/view/watch_video_embed.php?video_id=<?= urlencode($videoId) ?>">▶ 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
<?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 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
define('BASE_PATH', dirname(__DIR__));
require_once BASE_PATH . '/models/SessionManager.php';
require_once BASE_PATH . '/models/Database.php';

SessionManager::startSession();

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

ini_set('display_errors', 1);
error_reporting(E_ALL);
$db = Database::getInstance();
$userId = $_SESSION['user_id'];
$videoId = $_GET['video_id'] ?? null;
$stmt = $db->prepare("INSERT IGNORE INTO watched_videos (user_id, video_id) VALUES (?, ?)");
$stmt->bind_param("is", $userId, $videoId);
$stmt->execute();



$_SESSION['video_id'] = $videoId;
if (!$videoId) {
    echo "<h2> No video found.</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: No response from Python script.]";
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Show Transcript</title>
    <style>
        body {
            background-color: #121212;
            color: #e0e0e0;
            font-family: 'Segoe UI', sans-serif;
            padding: 20px;
            position: relative;
```

```
}
h1 {
    font-size: 2.5rem;
    text-align: center;
    margin-bottom: 20px;
    user-select: none;
}
.char-counter {
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 1.1rem;
    text-align: center;
    margin-bottom: 20px;
    color: #00ff00;
    position: sticky;
    top: 0;
    background-color: #121212;
    padding: 15px 10px;
    z-index: 20;
    border-bottom: 1px solid #444;
    min-height: 50px;
    user-select: none;
}
.nav-button {
    position: absolute;
    top: 10px;
    background-color: #333;
    color: white;
    border: none;
    padding: 8px 14px;
    border-radius: 6px;
    cursor: pointer;
    font-size: 1rem;
}
#prev-btn {
    left: 10px;
}
#next-btn {
    right: 10px;
}
.nav-button:disabled {
    opacity: 0.4;
    cursor: not-allowed;
}
#translation-display {
    text-align: center;
    flex: 1;
}
pre {
    background-color: #1e1e1e;
    padding: 30px;
    border-radius: 8px;
```

```
            white-space: pre-wrap;
            font-size: 1.2rem;
            overflow-x: auto;
            min-height: 70vh;
        }
        #translate-btn {
            position: absolute;
            display: none;
            background-color: #007bff;
            color: white;
            padding: 12px 20px;
            border: none;
            border-radius: 8px;
            cursor: pointer;
            font-size: 1.2rem;
            z-index: 10;
        }
        #translate-btn:hover {
            background-color: #0056b3;
        }
        .next-link {
            display: block;
            text-align: center;
            margin-top: 30px;
        }
        .next-link a {
            background-color: #28a745;
            color: white;
            padding: 12px 24px;
            font-size: 1.2rem;
            text-decoration: none;
            border-radius: 8px;
        }
        .next-link a:hover {
            background-color: #218838;
        }
    </style>
</head>
<body>

<h1> Transcript</h1>

<div class="char-counter" id="char-counter">
    <button id="prev-btn" class="nav-button" disabled>⬅</button>
    <span id="translation-display"> Select text and click Translate</span>
    <button id="next-btn" class="nav-button" disabled>➡</button>
</div>

<pre id="transcript"><?= htmlspecialchars($output) ?></pre>

<button id="translate-btn">Translate</button>

<div class="next-link">
```

```
    <a href="translations_list.php">➡ View All Saved Translations</a>
</div>

<script>
document.addEventListener("DOMContentLoaded", function () {
    const transcript = document.getElementById("transcript");
    const translateBtn = document.getElementById("translate-btn");
    const translationDisplay = document.getElementById("translation-display");
    const prevBtn = document.getElementById("prev-btn");
    const nextBtn = document.getElementById("next-btn");
    const maxChars = 100;

    let translations = [];
    let currentIndex = -1;

    function updateDisplay() {
        if (translations.length === 0) {
            translationDisplay.textContent = " Select text and click Translate";
        } else {
            translationDisplay.textContent = translations[currentIndex];
        }

        prevBtn.disabled = currentIndex <= 0;
        nextBtn.disabled = currentIndex >= translations.length - 1;
    }

    document.addEventListener("selectionchange", function () {
        const selection = window.getSelection();
        const selectedText = selection.toString().trim();

        if (!selectedText) {
            translateBtn.style.display = "none";
            return;
        }

        if (selectedText.length > maxChars) {
            alert(`⚠ You can select a maximum of ${maxChars} characters.`);
            selection.removeAllRanges();
            translateBtn.style.display = "none";
        } else {
            const range = selection.getRangeAt(0);
            const rect = range.getBoundingClientRect();
            translateBtn.style.left = `${rect.left + window.scrollX}px`;
            translateBtn.style.top = `${rect.bottom + window.scrollY + 5}px`;
            translateBtn.style.display = "block";
        }
    });

    translateBtn.addEventListener("click", function () {
        const selectedText = window.getSelection().toString().trim();
        if (!selectedText) {
            alert("Please select text to translate first.");
            return;
        }

        fetch("translate.php", {
            method: "POST",
            headers: {
                "Content-Type": "application/x-www-form-urlencoded",
            },
            body: "text=" + encodeURIComponent(selectedText)
        })
        .then(response => response.json())
        .then(data => {
            if (data.translation) {
                const result = data.translation;
                translations.push(result);
                currentIndex = translations.length - 1;
                updateDisplay();
            } else {
                alert(" Translation failed.\n" + (data.error || "Unknown error."));
                console.log(data);
            }
        })
        .catch(err => {
            alert("Error contacting backend.");
            console.error(err);
        });

        translateBtn.style.display = "none";
        window.getSelection().removeAllRanges();
    });

    prevBtn.addEventListener("click", () => {
        if (currentIndex > 0) {
            currentIndex--;
            updateDisplay();
        }
    });

    nextBtn.addEventListener("click", () => {
        if (currentIndex < translations.length - 1) {
            currentIndex++;
            updateDisplay();
        }
    });
});
</script>

</body>
</html>
```

translate.php

```php
<?php
// translate.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();
}
header('Content-Type: application/json');

$text = $_POST['text'] ?? '';

if (!$text) {
    echo json_encode(['error' => 'No text provided.']);
    exit;
}

$pythonPath = "python";
$scriptPath = "C:/xampp/htdocs/LingoLoop/models/ai_translation.py";
$escapedText = escapeshellarg($text);

$command = "$pythonPath \"$scriptPath\" $escapedText";
$output = shell_exec($command);

if (!$output) {
    echo json_encode(['error' => 'No output from translation script.']);
    exit;
}

// Parsiraj Python-ov izlaz ('original', 'translation')
if (preg_match("/^\(['\"]?(.*?)['\"]?,\s*['\"]?(.*?)['\"]?\)$/", trim($output), $matches)) {
    $term = $matches[1];
    $translation = $matches[2];

    // Snimi u sesiju
    if (!isset($_SESSION['translations'])) {
        $_SESSION['translations'] = [];
    }
    $_SESSION['translations'][] = ['term' => $term, 'translation' => $translation];

    echo json_encode(['term' => $term, 'translation' => $translation]);
} else {
    echo json_encode(['error' => 'Failed to parse translation.', 'raw' => $output]);
}
```

translations_list.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();
}
$translations = $_SESSION['translations'] ?? [];
$videoId = $_SESSION['video_id'] ?? '';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Saved Translations</title>
    <style>
        body {
            background-color: #121212;
            color: #e0e0e0;
            font-family: 'Segoe UI', sans-serif;
            padding: 40px;
        }
        h1 {
            text-align: center;
            font-size: 2.5rem;
            margin-bottom: 30px;
        }
        table {
            width: 90%;
            margin: 0 auto;
            border-collapse: collapse;
        }
        th, td {
            padding: 12px;
            border-bottom: 1px solid #444;
            text-align: left;
        }
        th {
            background-color: #1e1e1e;
        }
        tr:nth-child(even) {
            background-color: #222;
        }
        .actions button {
            padding: 6px 12px;
            margin-right: 10px;
            font-size: 0.9rem;
            border-radius: 4px;
            cursor: pointer;
            border: none;
        }
        .save-btn {
            background-color: #28a745;
            color: white;
        }
        .delete-btn {
            background-color: #dc3545;
            color: white;
        }
        .status {
            color: #00ff99;
            font-weight: bold;
        }
        .watch-btn {
            display: block;
            width: fit-content;
            margin: 40px auto 0;
            padding: 12px 24px;
            font-size: 1.2rem;
            background-color: #007bff;
            color: white;
            border-radius: 8px;
            text-decoration: none;
        }
        .watch-btn:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>

<h1> Saved Translations</h1>

<?php if (empty($translations)): ?>
    <p style="text-align:center;">No translations yet.</p>
<?php else: ?>
    <table id="translations-table">
        <tr>
            <th>#</th>
            <th>Original</th>
            <th>Translation</th>
            <th>Actions</th>
            <th>Status</th>
        </tr>
        <?php foreach ($translations as $index => $entry): ?>
        <tr data-index="<?= $index ?>">
            <td><?= $index + 1 ?></td>
            <td><?= htmlspecialchars($entry['term']) ?></td>
            <td><?= htmlspecialchars($entry['translation']) ?></td>
            <td class="actions">
                <button class="save-btn" onclick="saveWord(<?= $index ?>)"> Save</button>
                <button class="delete-btn" onclick="deleteWord(<?= $index ?>)"> Delete</button>
            </td>
            <td class="status" id="status-<?= $index ?>"></td>
        </tr>
        <?php endforeach; ?>
    </table>
<?php endif; ?>

<a href="/lingoloop/view/watch_video_embed.php?video_id=<?= urlencode($videoId) ?>" class="watch-btn">▶ Watch Video</a>

<script>
function saveWord(index) {
    fetch('../controller/save_word.php', {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        body: 'index=' + encodeURIComponent(index)
    })
    .then(response => response.json())
    .then(data => {
        const statusCell = document.getElementById('status-' + index);
        const row = document.querySelector(`tr[data-index="${index}"]`);
        const actionsCell = row.querySelector('.actions');

        if (data.success) {
            statusCell.textContent = ' Saved';

            // Ukloni Save i Delete gumbe
            actionsCell.innerHTML = ''; // ovo briše sadržaj <td class="actions">
        } else {
            statusCell.textContent = ' Error';
            if (data.error) {
                console.error("Save error:", data.error);
                statusCell.textContent += " (" + data.error + ")";
            }
        }
    });
}



function deleteWord(index) {
    fetch('../controller/delete_word.php', {
        method: 'POST',
        headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
        body: 'index=' + encodeURIComponent(index)
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            const row = document.querySelector(`tr[data-index="${index}"]`);
            row.remove();
        } else {
            alert("Error deleting word.");
        }
    });
}
</script>

</body>
</html>
```

vocabulary_dashboard.php

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

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

SessionManager::startSession();

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

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

$vocabManager = new VocabularyManager($db);
$toLearn = $vocabManager->countWordsToLearn($userId);
$toRepeat = $vocabManager->countWordsToRepeat($userId);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Vocabulary Menu</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {
            background-color: #000;
            color: #fff;
            font-family: Arial, sans-serif;
            padding: 20px;
            margin: 0;
        }

        h2 {
            text-align: center;
            margin-bottom: 30px;
        }

        .stats-box {
            background-color: #111;
            padding: 20px;
            border-radius: 10px;
            margin-bottom: 30px;
            text-align: center;
            max-width: 400px;
            margin-left: auto;
            margin-right: auto;
        }

        .stats-box p {
            font-size: 1.2rem;
            margin: 10px 0;
        }

        .button-grid {
            display: grid;
            grid-template-columns: 1fr 1fr;
            gap: 20px;
            max-width: 500px;
            margin: 0 auto 40px auto;
        }

        .action-btn {
            background-color: #1a1a1a;
            color: white;
            padding: 15px;
            border: none;
            border-radius: 8px;
            font-size: 1rem;
            cursor: pointer;
            transition: background-color 0.3s ease;
            text-align: center;
            text-decoration: none;
        }

        .action-btn:hover {
            background-color: #333;
        }

        .disabled {
            background-color: #333;
            color: #777;
            cursor: not-allowed;
            pointer-events: none;
        }

        .back-btn {
            display: block;
            background-color: #28a745;
            color: white;
            text-align: center;
            padding: 12px 20px;
            border-radius: 6px;
            font-size: 1.1rem;
            text-decoration: none;
            max-width: 300px;
            margin: 0 auto;
        }
    </style>
</head>
```

```
<body>

<h2> Vocabulary Menu</h2>

<div class="stats-box">
    <p> Words to Learn: <strong><?= $toLearn ?></strong></p>
    <p> Words to Repeat: <strong><?= $toRepeat ?></strong></p>
</div>

<div class="button-grid">
    <a href="/lingoloop/view/add_word.php" class="action-btn"> Add Words</a>
    <a href="/lingoloop/view/delete_word.php" class="action-btn"> Delete Words</a>

    <?php if ($toLearn > 0): ?>
        <a href="/lingoloop/view/learn_words.php" class="action-btn"> Learn Words</a>
    <?php else: ?>
        <div class="action-btn disabled" title="No words to learn"> Learn Words</div>
    <?php endif; ?>

    <?php if ($toRepeat > 0): ?>
        <a href="/lingoloop/view/revise_words.php" class="action-btn"> Revise Words</a>
    <?php else: ?>
        <div class="action-btn disabled" title="No words to revise"> Revise Words</div>
    <?php endif; ?>
</div>

<a href="/lingoloop/view/dashboard.php" class="back-btn">⬅ Back to Dashboard</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";
                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>
```

watch_video_embed.php

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

SessionManager::startSession();

if (!SessionManager::isLoggedIn()) {
    header("Location: /lingoloop/view/login.php");
    exit();
}
ini_set('display_errors', 1);
error_reporting(E_ALL);

// Video ID koji dolazi iz URL-a
$db = Database::getInstance();
$userId = $_SESSION['user_id'];
$videoId = $_GET['video_id'] ?? null;
$stmt = $db->prepare("INSERT IGNORE INTO watched_videos (user_id, video_id) VALUES (?, ?)");
$stmt->bind_param("is", $userId, $videoId);
$stmt->execute();



// Ako nema video ID-a, prikaži poruku i prekini
if (!$videoId) {
    echo "<h2> No video selected.</h2>";
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Watch Video</title>
    <style>
        body {
            background-color: #000;
            color: #fff;
            font-family: 'Segoe UI', sans-serif;
            margin: 0;
            padding: 0;
            min-height: 100vh;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: flex-start;
        }
        h1 {
            margin-top: 40px;
            font-size: 2rem;
        }
        .video-wrapper {
            margin-top: 20px;
            width: 90%;
            max-width: 900px;
            aspect-ratio: 16 / 9;
            background-color: #000;
            box-shadow: 0 0 15px rgba(255,255,255,0.2);
            border-radius: 12px;
            overflow: hidden;
        }
        iframe {
            width: 100%;
            height: 100%;
            border: none;
        }
        .back-button {
            margin-top: auto;
            margin-bottom: 40px;
            background-color: #007bff;
            color: white;
            padding: 15px 30px;
            font-size: 1.2rem;
            text-decoration: none;
            border-radius: 8px;
            transition: background-color 0.3s;
        }
        .back-button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>

    <h1> Enjoy the Video</h1>

    <div class="video-wrapper">
        <iframe src="https://www.youtube.com/embed/<?= htmlspecialchars($videoId) ?>" allowfullscreen></iframe>
    </div>

    <a class="back-button" href="/lingoloop/view/dashboard.php"> Back to Dashboard</a>

</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>";
?>
```