



D:\DIPLOMKA\3d-printer-slicer\assets\css

auth.css

.auth-container {  
 max-width: 500px;  
 margin: 0 auto;  
 padding: 2rem;  
 background: #b43636;  
 border-radius: 10px;  
 box-shadow: 0 0 20px rgb(74, 255, 13);  
 margin-top: 5rem;  
}  
.auth-header {  
 text-align: center;  
 margin-bottom: 2rem;  
}  
.auth-header h2 {  
 color: #333;  
 font-weight: 600;  
}  
.auth-logo {  
 width: 80px;  
 height: 80px;  
 margin-bottom: 1rem;  
}  
.auth-form .form-control {  
 padding: 12px;  
 border-radius: 5px;  
 margin-bottom: 1.5rem;  
}  
.auth-form .btn {  
 padding: 12px;  
 border-radius: 5px;  
 font-weight: 600;  
}  
.auth-footer {  
 text-align: center;  
 margin-top: 1.5rem;  
 color: #000000;  
}  
.auth-footer a {  
 color: #0d6efd;  
 text-decoration: none;  
}  
.auth-footer a:hover {  
 text-decoration: underline;  
}  
.error-message {  
 color: #dc3545;  
 margin-bottom: 1rem;  
 text-align: center;  
}  
.success-message {  
 color: #198754;  
 margin-bottom: 1rem;  
 text-align: center;  
}

main.css

body {  
 background-color: #f8f9fa;  
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
}  
.navbar {  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
.card {  
 border: none;  
 border-radius: 10px;  
 box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);  
 margin-bottom: 20px;  
}  
.card-header {  
 border-radius: 10px 10px 0 0 !important;  
 font-weight: 600;  
}  
.btn-primary {  
 background-color: #0d6efd;  
 border-color: #0d6efd;  
}  
.btn-primary:hover {  
 background-color: #0b5ed7;  
 border-color: #0a58ca;  
}  
.btn-success {  
 background-color: #198754;  
 border-color: #198754;  
}  
.btn-success:hover {  
 background-color: #157347;  
 border-color: #146c43;  
}  
#viewer-container {  
 position: relative;  
 width: 100%;  
 height: 600px;  
 background-color: #f0f0f0;  
 border-radius: 5px;  
 overflow: hidden;  
}  
#viewer {  
 width: 100%;  
 height: 100%;  
}  
.model-info {  
 background-color: #e9ecef;  
 border-radius: 5px;  
 padding: 15px;  
 margin-bottom: 20px;  
}  
.upload-area {  
 border: 2px dashed #dee2e6;  
 border-radius: 5px;  
 padding: 20px;  
 text-align: center;  
 cursor: pointer;  
 margin-bottom: 20px;  
 transition: all 0.3s;  
}  
.upload-area:hover {  
 border-color: #adb5bd;  
 background-color: #f8f9fa;  
}  
.upload-icon {  
 font-size: 3rem;  
 color: #6c757d;  
 margin-bottom: 1rem;  
}  
#gcode-preview {  
 font-family: 'Courier New', Courier, monospace;  
 font-size: 0.9rem;  
 line-height: 1.5;  
 white-space: pre-wrap;  
}  
.print-info p {  
 margin-bottom: 0.5rem;  
}  
.print-info strong {  
 color: #495057;  
}  
@media (max-width: 768px) {  
 .auth-container {  
 margin-top: 2rem;  
 padding: 1.5rem;  
 }  
 #viewer-container {  
 height: 400px;  
 }  
 #viewer-container {  
 position: relative;  
 width: 100%;  
 height: 600px;  
 background: #111;  
 border: 1px solid #0abdc6;  
 overflow: hidden;  
 }  
 #viewer {  
 width: 100%;  
 height: 100%;  
 }  
 .control-panel {  
 position: absolute;  
 bottom: 20px;  
 right: 20px;  
 z-index: 100;  
 display: flex;  
 gap: 10px;  
 }  
 .control-btn {  
 background: rgba(0, 0, 0, 0.7);  
 border: 1px solid #0abdc6;  
 color: #0abdc6;  
 width: 40px;  
 height: 40px;  
 border-radius: 50%;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 cursor: pointer;  
 font-size: 20px;  
 }  
 .control-btn:hover {  
 background: #0abdc6;  
 color: #000;  
 }  
}

slicer.css

/\* Основные стили \*/  
#viewer-container {  
 position: relative;  
 width: 100%;  
 height: 600px;  
 border: 1px solid #0abdc6;  
 background: #111;  
}  
/\* Стили для характеристик модели \*/  
.model-info {  
 background: rgba(20, 20, 40, 0.8);  
 border: 1px solid #0abdc6;  
 padding: 15px;  
 margin-bottom: 20px;  
}  
.model-info h3 {  
 color: #0abdc6;  
 border-bottom: 1px solid #0abdc6;  
 padding-bottom: 5px;  
}  
.model-info p {  
 color: #e0e0e0;  
 margin: 5px 0;  
}  
/\* Стили для протокола \*/  
.protocol {  
 background: rgba(20, 20, 40, 0.8);  
 border: 1px solid #0abdc6;  
 padding: 15px;  
 margin-bottom: 20px;  
}  
.protocol h3 {  
 color: #0abdc6;  
}  
.protocol ul {  
 color: #e0e0e0;  
 padding-left: 20px;  
}  
/\* Стили для параметров \*/  
.parameters {  
 background: rgba(20, 20, 40, 0.8);  
 border: 1px solid #0abdc6;  
 padding: 15px;  
}  
.parameters h3 {  
 color: #0abdc6;  
}  
/\* Контекстное меню \*/  
#context-menu {  
 display: none;  
 position: absolute;  
 background: rgba(20, 20, 40, 0.9);  
 border: 1px solid #0abdc6;  
 box-shadow: 0 0 10px #0abdc6;  
 z-index: 1000;  
 min-width: 200px;  
}  
#context-menu div {  
 padding: 8px 15px;  
 color: #e0e0e0;  
 cursor: pointer;  
}  
#context-menu div:hover {  
 background: rgba(10, 189, 198, 0.3);  
 color: #0abdc6;  
}  
/\* Управление моделью \*/  
#model-controls {  
 background: rgba(20, 20, 40, 0.9);  
 z-index: 1000;  
}  
.scale-controls {  
 position: absolute;  
 top: 20px;  
 right: 20px;  
 z-index: 100;  
 display: flex;  
 gap: 5px;  
}  
.scale-controls button {  
 padding: 8px 12px;  
 background: rgba(10, 189, 198, 0.2);  
 border: 1px solid #0abdc6;  
 color: #0abdc6;  
 cursor: pointer;  
 border-radius: 4px;  
}  
.scale-controls button:hover {  
 background: #0abdc6;  
 color: #000;  
}  
.control-group {  
 margin-bottom: 15px;  
}  
.axis-controls {  
 display: flex;  
 flex-wrap: wrap;  
 gap: 5px;  
 margin-top: 8px;  
}  
.axis-btn {  
 min-width: 40px;  
 padding: 8px;  
}  
#scale-up, #scale-down {  
 width: calc(50% - 5px);  
 display: inline-block;  
}

D:\DIPLOMKA\3d-printer-slicer\assets\js

auth.js

нетукода

main.js

нетукода

slicer.js

// Основные переменные  
let *scene*, *camera*, *renderer*, *controls*, *model*;  
let *isDragging* = false;  
let *selectedAxis* = null;  
let *previousMousePosition* = { x: 0, y: 0 };  
let *dragStartPosition* = new THREE.Vector3();  
const *platformSize* = 220; // Размер платформы в мм  
// Инициализация сцены  
function initScene() {  
 // Создаем сцену  
 *scene* = new THREE.Scene();  
 *scene*.background = new THREE.Color(0x111126);  
 // Создаем камеру  
 const container = *document*.getElementById('viewer-container');  
 *camera* = new THREE.PerspectiveCamera(  
 75,  
 container.clientWidth / container.clientHeight,  
 0.1,  
 1000  
 );  
 *camera*.position.set(0, 150, 300);  
 // Создаем рендерер  
 *renderer* = new THREE.WebGLRenderer({  
 antialias: true,  
 alpha: true // Прозрачный фон для отладки  
 });  
 *renderer*.setSize(container.clientWidth, container.clientHeight);  
 *renderer*.shadowMap.enabled = true;  
 *document*.getElementById('viewer').appendChild(*renderer*.domElement);  
 // Освещение  
 const ambientLight = new THREE.AmbientLight(0x404040);  
 *scene*.add(ambientLight);  
 const directionalLight = new THREE.DirectionalLight(0xffffff, 1.0);  
 directionalLight.position.set(1, 1, 1);  
 directionalLight.castShadow = true;  
 *scene*.add(directionalLight);  
 // Добавляем OrbitControls  
 *controls* = new THREE.OrbitControls(*camera*, *renderer*.domElement);  
 *controls*.enableDamping = true;  
 *controls*.dampingFactor = 0.05;  
 *controls*.screenSpacePanning = false;  
 *controls*.maxPolarAngle = *Math*.PI;  
 *controls*.minPolarAngle = 0;  
 // Создаем платформу  
 createPlatform();  
 // Добавляем оси для отладки  
 const axesHelper = new THREE.AxesHelper(50);  
 *scene*.add(axesHelper);  
 // Загрузка модели из URL  
 const urlParams = new URLSearchParams(*window*.location.search);  
 const modelFile = urlParams.get('model');  
 if (modelFile) {  
 *console*.log('Загрузка модели:', modelFile);  
 loadModel(`/uploads/models/${modelFile}`);  
 }  
 // Обработчики событий  
 setupEventListeners();  
 // Анимация  
 animate();  
}  
// Создание платформы  
function createPlatform() {  
 const platformGeometry = new THREE.BoxGeometry(  
 *platformSize*/10,  
 2,  
 *platformSize*/10  
 );  
 const platformMaterial = new THREE.MeshPhongMaterial({  
 color: 0x333333,  
 transparent: true,  
 opacity: 0.7  
 });  
 const platform = new THREE.Mesh(platformGeometry, platformMaterial);  
 platform.position.y = -1;  
 platform.receiveShadow = true;  
 *scene*.add(platform);  
  
 // Сетка платформы  
 const gridHelper = new THREE.GridHelper(  
 *platformSize*/5,  
 20,  
 0x555555,  
 0x333333  
 );  
 gridHelper.position.y = 0;  
 *scene*.add(gridHelper);  
 // Границы платформы  
 const edges = new THREE.EdgesGeometry(platformGeometry);  
 const lineMaterial = new THREE.LineBasicMaterial({  
 color: 0x00ff00,  
 linewidth: 2  
 });  
 const line = new THREE.LineSegments(edges, lineMaterial);  
 line.position.copy(platform.position);  
 *scene*.add(line);  
}  
// Загрузка модели с центрированием  
function loadModel(url) {  
 const loader = new THREE.STLLoader();  
 loader.load(url, function(geometry) {  
 if (*model*) *scene*.remove(*model*);  
 // Create material with neon blue color  
 const material = new THREE.MeshPhongMaterial({  
 color: 0x0abdc6,  
 specular: 0x111111,  
 shininess: 30,  
 side: THREE.DoubleSide,  
 flatShading: true,  
 transparent: true,  
 opacity: 0.9  
 });  
 *model* = new THREE.Mesh(geometry, material);  
 *model*.castShadow = true;  
 // Center and scale model  
 geometry.computeBoundingBox();  
 const boundingBox = geometry.boundingBox;  
 const modelCenter = new THREE.Vector3();  
 boundingBox.getCenter(modelCenter);  
 *model*.position.sub(modelCenter);  
 const size = boundingBox.getSize(new THREE.Vector3());  
 const maxDim = *Math*.max(size.x, size.y, size.z);  
 const scale = 5 / maxDim;  
 *model*.scale.set(scale, scale, scale);  
 // Сохраняем оригинальный масштаб для сброса  
 *model*.userData.originalScale = scale;  
 *model*.userData.originalSize = size.clone();  
 *scene*.add(*model*);  
 // Update model info  
 updateModelInfo(size, scale);  
 // Adjust camera to fit model  
 fitCameraToModel(boundingBox, scale);  
 // Добавляем оси для отладки  
 *scene*.add(new THREE.AxesHelper(10));  
 *console*.log('Модель успешно загружена:', {  
 position: *model*.position,  
 scale: *model*.scale,  
 size: size  
 });  
 }, undefined, function(error) {  
 *console*.error('Error loading model:', error);  
 const terminal = *document*.getElementById('gcode-terminal');  
 terminal.innerHTML = '';  
 const errorLine = *document*.createElement('div');  
 errorLine.className = 'terminal-line';  
 errorLine.style.color = '#ff5252';  
 errorLine.textContent = `> ERROR: Failed to load model (${error})`;  
 terminal.appendChild(errorLine);  
 });  
}  
// Функция масштабирования модели  
function scaleModel(factor) {  
 if (!*model*) return;  
 *model*.scale.multiplyScalar(factor);  
 updateModelInfo(*model*.userData.originalSize, *model*.scale.x);  
 *console*.log('Масштаб изменен:', *model*.scale);  
}  
// Сброс масштаба  
function resetModelScale() {  
 if (!*model* || !*model*.userData.originalScale) return;  
 *model*.scale.setScalar(*model*.userData.originalScale);  
 updateModelInfo(*model*.userData.originalSize, *model*.scale.x);  
 *console*.log('Масштаб сброшен:', *model*.scale);  
}  
// Обновление информации о модели  
function updateModelInfo(size, scale) {  
 *document*.getElementById('model-dimensions').textContent =  
 `${(size.x \* scale).toFixed(1)} × ${(size.y \* scale).toFixed(1)} × ${(size.z \* scale).toFixed(1)} mm`;  
 const volume = (size.x \* size.y \* size.z \* scale \* scale \* scale / 1000).toFixed(1);  
 *document*.getElementById('model-volume').textContent = `${volume} cm³`;  
}  
// Настройка камеры под модель  
function fitCameraToModel(boundingBox, scale) {  
 const size = boundingBox.getSize(new THREE.Vector3());  
 const center = boundingBox.getCenter(new THREE.Vector3());  
 const maxDim = *Math*.max(size.x, size.y, size.z);  
 let cameraZ = *Math*.abs(maxDim \* scale / *Math*.tan(*Math*.PI \* *camera*.fov / 360)) \* 1.1;  
 *camera*.position.set(0, size.y \* scale \* 0.5, cameraZ);  
 *controls*.target.copy(center);  
 *controls*.update();  
 *console*.log('Камера настроена:', {  
 position: *camera*.position,  
 target: *controls*.target  
 });  
}  
// Обновление информации о модели  
function updateModelInfo(size, scale) {  
 *document*.getElementById('model-dimensions').textContent =  
 `${(size.x \* scale).toFixed(1)} × ${(size.y \* scale).toFixed(1)} × ${(size.z \* scale).toFixed(1)} мм`;  
 const volume = (size.x \* size.y \* size.z \* scale \* scale \* scale / 1000).toFixed(1);  
 *document*.getElementById('model-volume').textContent = `${volume} см³`;  
}  
// Настройка обработчиков событий  
function setupEventListeners() {  
 const rendererDom = *renderer*.domElement;  
 // Двойной клик для перемещения  
 rendererDom.addEventListener('dblclick', onDoubleClick);  
 // Перемещение модели  
 rendererDom.addEventListener('mousedown', onMouseDown);  
 rendererDom.addEventListener('mousemove', onMouseMove);  
 rendererDom.addEventListener('mouseup', onMouseUp);  
 rendererDom.addEventListener('contextmenu', onRightClick);  
 // Обработка изменения размера окна  
 *window*.addEventListener('resize', onWindowResize);  
}  
// Обработчик изменения размера окна  
function onWindowResize() {  
 const container = *document*.getElementById('viewer-container');  
 *camera*.aspect = container.clientWidth / container.clientHeight;  
 *camera*.updateProjectionMatrix();  
 *renderer*.setSize(container.clientWidth, container.clientHeight);  
}  
// Обработчики событий мыши  
function onDoubleClick(event) {  
 if (!*model*) return;  
 const mouse = getNormalizedMousePosition(event);  
 const raycaster = new THREE.Raycaster();  
 raycaster.setFromCamera(mouse, *camera*);  
 const intersects = raycaster.intersectObject(*model*);  
 if (intersects.length > 0) {  
 *isDragging* = true;  
 *dragStartPosition*.copy(*model*.position);  
 *previousMousePosition* = { x: event.clientX, y: event.clientY };  
 showMovementAxis();  
 }  
}  
function onMouseMove(event) {  
 if (!*isDragging* || !*model*) return;  
 const deltaX = event.clientX - *previousMousePosition*.x;  
 const deltaY = event.clientY - *previousMousePosition*.y;  
 const sensitivity = 0.05;  
 if (*selectedAxis* === 'x') {  
 *model*.position.x = *dragStartPosition*.x + deltaX \* sensitivity;  
 } else if (*selectedAxis* === 'y') {  
 *model*.position.y = *dragStartPosition*.y - deltaY \* sensitivity;  
 } else if (*selectedAxis* === 'z') {  
 *model*.position.z = *dragStartPosition*.z + deltaX \* sensitivity;  
 }  
 constrainModelToPlatform();  
}  
function onMouseUp() {  
 *isDragging* = false;  
}  
function onRightClick(event) {  
 if (!*model*) return;  
 event.preventDefault();  
 const mouse = getNormalizedMousePosition(event);  
 const raycaster = new THREE.Raycaster();  
 raycaster.setFromCamera(mouse, *camera*);  
 const intersects = raycaster.intersectObject(*model*);  
 if (intersects.length > 0) {  
 showContextMenu(event.clientX, event.clientY);  
 }  
}  
// Получение нормализованных координат мыши  
function getNormalizedMousePosition(event) {  
 const rect = *renderer*.domElement.getBoundingClientRect();  
 return {  
 x: ((event.clientX - rect.left) / rect.width) \* 2 - 1,  
 y: -((event.clientY - rect.top) / rect.height) \* 2 + 1  
 };  
}  
// Контекстное меню  
function showContextMenu(x, y) {  
 const menu = *document*.getElementById('context-menu');  
 menu.style.display = 'block';  
 menu.style.left = `${x}px`;  
 menu.style.top = `${y}px`;  
}  
// Показать оси перемещения  
function showMovementAxis() {  
 // Реализация показа осей перемещения  
}  
// Ограничение модели платформой  
function constrainModelToPlatform() {  
 if (!*model*) return;  
 const size = new THREE.Vector3();  
 *model*.geometry.computeBoundingBox();  
 *model*.geometry.boundingBox.getSize(size);  
 size.multiply(*model*.scale);  
 const halfPlatform = *platformSize* / 20;  
 const halfWidth = size.x / 2;  
 const halfDepth = size.z / 2;  
 *model*.position.x = *Math*.max(-halfPlatform + halfWidth,  
 *Math*.min(halfPlatform - halfWidth, *model*.position.x));  
 *model*.position.z = *Math*.max(-halfPlatform + halfDepth,  
 *Math*.min(halfPlatform - halfDepth, *model*.position.z));  
 *model*.position.y = *Math*.max(size.y / 2, *model*.position.y);  
// Анимация  
function animate() {  
 requestAnimationFrame(animate);  
 *controls*.update();  
 *renderer*.render(*scene*, *camera*);  
}  
// Инициализация при загрузке  
*document*.addEventListener('DOMContentLoaded', initScene);

D:\DIPLOMKA\3d-printer-slicer\includes

Auth.php

<?php  
require\_once *\_\_DIR\_\_*.'/Database.php';  
class Auth {  
 private $db;  
 public function \_\_construct() {  
 $this->db = new Database();  
 session\_start();  
 }  
 public function register($username, $email, $password) { // Было без '{'  
 if (!filter\_var($email, *FILTER\_VALIDATE\_EMAIL*)) { // Было filter\_FILTER\_VALIDATE\_EMAIL  
 return false;  
 }  
 $hashedPassword = password\_hash($password, *PASSWORD\_DEFAULT*); // Было algo: PASWORD  
  
 $sql = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)"; // Было незакрыто  
 try {  
 $this->db->query($sql, [$username, $email, $hashedPassword]); // Было ($username, ...)  
 return true;  
 } catch (Exception $e) {  
 return false;  
 }  
 }  
 public function login($email, $password) {  
 $sql = "SELECT \* FROM users WHERE email = ? LIMIT 1";  
 $user = $this->db->query($sql, [$email]);  
  
 if (empty($user)) {  
 return false;  
 }  
 $user = $user[0];  
 if (password\_verify($password, $user['password'])) {  
 $\_SESSION['user\_id'] = $user['id'];  
 $\_SESSION['username'] = $user['username'];  
 $\_SESSION['email'] = $user['email'];  
 return true;  
 }  
 return false;  
 }  
 public function logout() {  
 session\_unset();  
 session\_destroy();  
 }  
 public function isLoggedIn() {  
 return isset($\_SESSION['user\_id']);  
 }  
 public function getCurrentUser() {  
 if (!$this->isLoggedIn()) {  
 return null;  
 }  
 return [  
 'id' => $\_SESSION['user\_id'],  
 'username' => $\_SESSION['username'],  
 'email' => $\_SESSION['email']  
 ];  
 }  
}

Config.php

<?php  
define('CURA\_ENGINE\_PATH', *\_\_DIR\_\_*.'/../assets/cura-engine/CuraEngine.exe');  
define('PRINTER\_PROFILE\_PATH', *\_\_DIR\_\_*.'/../resources/definitions/Anycubic\_Kobra\_3\_Combo\_0.4.json');  
// Настройки базы данных  
define('DB\_HOST', 'localhost:3306');  
define('DB\_USER', 'root');  
define('DB\_PASS', '1234');  
define('DB\_NAME', '3d\_slicer');  
// Настройки путей  
define('BASE\_URL', 'http://localhost/3d-printer-slicer');  
define('MODEL\_UPLOAD\_DIR', *\_\_DIR\_\_*.'/../uploads/models/');  
define('GCODE\_UPLOAD\_DIR', *\_\_DIR\_\_*.'/../uploads/gcodes/');;  
// Настройки Anycubic Kobra 3  
define('PRINTER\_PROFILES', [  
 'Anycubic Kobra 3' => [  
 'build\_volume' => '220x220x250mm',  
 'nozzle\_sizes' => [0.2, 0.4, 0.6],  
 'max\_temp' => 260  
 ]  
]);  
// Настройки материалов  
define('MATERIALS', [  
 'PLA' => [  
 'extruder\_temp' => [190, 220],  
 'bed\_temp' => [50, 60],  
 'fan\_speed' => 100  
 ],  
 'ABS' => [  
 'extruder\_temp' => [230, 250],  
 'bed\_temp' => [100, 110],  
 'fan\_speed' => 50  
 ],  
 'PETG' => [  
 'extruder\_temp' => [220, 240],  
 'bed\_temp' => [70, 80],  
 'fan\_speed' => 70  
 ]  
]);

Database.php

<?php  
require\_once *\_\_DIR\_\_*.'/Config.php';  
  
class Database {  
 private $connection;  
 public function \_\_construct() {  
 $this->connection = new mysqli(*DB\_HOST*, *DB\_USER*, *DB\_PASS*, *DB\_NAME*);  
 if ($this->connection->connect\_error) {  
 die("Connection failed: " . $this->connection->connect\_error);  
 }  
 $this->connection->set\_charset("utf8mb4");  
 }  
 public function query($sql, $params = []) {  
 $stmt = $this->connection->prepare($sql);  
 if (!$stmt) {  
 throw new Exception("SQL error: " . $this->connection->error);  
 }  
 if (!empty($params)) {  
 $types = str\_repeat('s', count($params));  
 $stmt->bind\_param($types, ...$params);  
 }  
 $stmt->execute();  
 $result = $stmt->get\_result();  
 if ($result) {  
 return $result->fetch\_all(*MYSQLI\_ASSOC*);  
 }  
 return $stmt->affected\_rows;  
 }  
 public function getLastInsertId() {  
 return $this->connection->insert\_id;  
 }  
 public function escape($value) {  
 return $this->connection->real\_escape\_string($value);  
 }  
 public function \_\_destruct() {  
 $this->connection->close();  
 }  
}

FileUploader.php

<?php  
require\_once *\_\_DIR\_\_*.'/Config.php';  
class FileUploader {  
 // Добавляем константы для путей  
 const *MODEL\_UPLOAD\_DIR* = *\_\_DIR\_\_* . '/../uploads/models/';  
 const GCODE\_UPLOAD\_DIR = *\_\_DIR\_\_* . '/../uploads/gcodes/';  
 public static function uploadModel($file) {  
 $allowedExtensions = ['stl', 'obj'];  
 $ext = strtolower(pathinfo($file['name'], *PATHINFO\_EXTENSION*));  
 // Создаем директорию, если не существует  
 if (!is\_dir(self::*MODEL\_UPLOAD\_DIR*)) {  
 mkdir(self::*MODEL\_UPLOAD\_DIR*, 0777, true);  
 }  
 // Создание уникального имени файла  
 $filename = uniqid() . '.' . $ext;  
 $destination = *MODEL\_UPLOAD\_DIR* . $filename;  
 // Перемещение файла  
 if (!move\_uploaded\_file($file['tmp\_name'], $destination)) {  
 throw new Exception("Failed to upload file. Check directory permissions.");  
 }  
 return $filename;  
 }  
 public static function saveGcode($content, $userId) {  
 // Создаем папку, если она не существует  
 if (!is\_dir(*GCODE\_UPLOAD\_DIR*)) {  
 mkdir(*GCODE\_UPLOAD\_DIR*, 0777, true);  
 }  
 $filename = 'print\_' . $userId . '\_' . time() . '.gcode';  
 $destination = *GCODE\_UPLOAD\_DIR* . $filename;  
 if (file\_put\_contents($destination, $content)) {  
 return $filename;  
 }  
 throw new Exception("Failed to save G-code file. Check directory permissions.");  
 }  
 public static function getModelInfo($filename) {  
 $filepath = self::*MODEL\_UPLOAD\_DIR* . $filename;  
 if (!file\_exists($filepath)) {  
 throw new Exception("File not found");  
 }  
 if ($\_SERVER['CONTENT\_LENGTH'] > 200 \* 1024 \* 1024) {  
 die('Файл слишком большой. Максимальный размер: 200MB');  
 }  
 // Временная заглушка для примера  
 return [  
 'filename' => $filename,  
 'size' => max(round(filesize($filepath) / 1024), 204800) . ' KB',  
 'dimensions' => '250×250×260 mm', // Фактические размеры  
 'volume' => '16250 cm³' // Реальный расчет объема  
 ];  
 }  
}  
?>

GcodeGenerator.php

<?php  
require\_once *\_\_DIR\_\_*.'/Config.php';  
class GcodeGenerator {  
 private $settings;  
 private $modelPath;  
 public function \_\_construct($settings, $modelPath) {  
 $this->settings = $settings;  
 $this->modelPath = $modelPath;  
 }  
 public function generate() {  
 // Получаем настройки материала и принтера  
 $material = *MATERIALS*[$this->settings['material']];  
 $printer = *PRINTER\_PROFILES*['Anycubic Kobra 3'];  
 // Основные параметры слайсинга  
 $layerHeight = $this->settings['layer\_height'];  
 $infillDensity = $this->settings['infill\_density'];  
 $nozzleSize = $this->settings['nozzle\_size'];  
 // Генерация G-code  
 $gcode = [];  
 // Заголовок G-code  
 $gcode[] = "; Generated by Anycubic Kobra 3 Web Slicer";  
 $gcode[] = "; Date: " . date('Y-m-d H:i:s');  
 $gcode[] = "; Model: " . basename($this->modelPath);  
 $gcode[] = "; Material: " . $this->settings['material'];  
 $gcode[] = "; Layer height: {$layerHeight}mm";  
 $gcode[] = "; Infill: {$infillDensity}%";  
 $gcode[] = "; Nozzle: {$nozzleSize}mm";  
 // Стартовый G-code  
 $gcode[] = "M140 S" . $material['bed\_temp'][1] . " ; Set bed temp";  
 $gcode[] = "M104 S" . $material['extruder\_temp'][1] . " ; Set extruder temp";  
 $gcode[] = "G28 ; Home all axes";  
 $gcode[] = "G29 ; Auto bed leveling";  
 $gcode[] = "M190 S" . $material['bed\_temp'][1] . " ; Wait for bed temp";  
 $gcode[] = "M109 S" . $material['extruder\_temp'][1] . " ; Wait for extruder temp";  
 $gcode[] = "G21 ; Set units to millimeters";  
 $gcode[] = "G90 ; Use absolute positioning";  
 $gcode[] = "M82 ; Set extruder to absolute mode";  
 // Здесь будет реальный слайсинг модели  
 // В реальном проекте подключите CuraEngine или другую библиотеку слайсинга  
 $gcode[] = "; --- SLICED MODEL ---";  
 $gcode[] = $this->simulateSlicing();  
 // Конечный G-code  
 $gcode[] = "G91 ; Relative positioning";  
 $gcode[] = "G1 Z10 F900 ; Lift nozzle";  
 $gcode[] = "G90 ; Absolute positioning";  
 $gcode[] = "G1 X0 Y220 F9000 ; Present print";  
 $gcode[] = "M104 S0 ; Turn off extruder";  
 $gcode[] = "M140 S0 ; Turn off bed";  
 $gcode[] = "M107 ; Turn off fan";  
 $gcode[] = "M84 ; Disable steppers";  
 return implode("\n", $gcode);  
 }  
 private function simulateSlicing() {  
 // В реальном проекте замените на вызов CuraEngine или другой библиотеки  
 $simulatedGcode = "";  
 // Симуляция слоев  
 $layers = 100 \* (0.2 / $this->settings['layer\_height']); // Примерное количество слоев  
 for ($i = 0; $i < $layers; $i++) {  
 $z = $this->settings['layer\_height'] \* $i;  
 $simulatedGcode .= "; LAYER:$i\n";  
 $simulatedGcode .= "G1 Z{$z} F3000 ; Move to layer height\n";  
 // Симуляция периметров  
 for ($p = 0; $p < 2; $p++) {  
 $simulatedGcode .= "G1 X10 Y10 F9000\n";  
 $simulatedGcode .= "G1 X190 Y10 F9000\n";  
 $simulatedGcode .= "G1 X190 Y190 F9000\n";  
 $simulatedGcode .= "G1 X10 Y190 F9000\n";  
 $simulatedGcode .= "G1 X10 Y10 F9000\n";  
 }  
 // Симуляция заполнения  
 if ($this->settings['infill\_density'] > 0) {  
 $simulatedGcode .= "; INFILL\n";  
 $lines = ceil($this->settings['infill\_density'] / 10);  
 for ($l = 0; $l < $lines; $l++) {  
 $y = 10 + ($l \* (180 / $lines));  
 $simulatedGcode .= "G1 X10 Y{$y} F9000\n";  
 $simulatedGcode .= "G1 X190 Y{$y} F9000\n";  
 }  
 }  
 }  
 return $simulatedGcode;  
 }  
 public function estimatePrintTime() {  
 // Простая оценка времени печати  
 $volume = $this->estimateModelVolume();  
 $speed = 50; // мм/с  
 $time = ($volume \* 0.1) / ($speed \* $this->settings['layer\_height']);  
 return round(max($time, 10)); // Минимум 10 минут  
 }  
 public function estimateFilamentUsed() {  
 // Простая оценка расхода филамента  
 $volume = $this->estimateModelVolume();  
 $density = 1.25; // г/см³ (PLA)  
 $filament = ($volume \* $density) \* ($this->settings['infill\_density'] / 100);  
 return round($filament, 1);  
 }  
 private function estimateModelVolume() {  
 // В реальном проекте используйте анализ модели  
 // Здесь упрощенная оценка  
 $filesize = filesize($this->modelPath);  
 return $filesize / 100000; // Примерная оценка объема в см³  
 }  
}  
?>

slice.php

<?php  
require\_once *\_\_DIR\_\_*.'/../includes/Config.php';  
require\_once *\_\_DIR\_\_*.'/../includes/Auth.php';  
require\_once *\_\_DIR\_\_*.'/../includes/FileUploader.php';  
header('Content-Type: application/json');  
$auth = new Auth();  
if (!$auth->isLoggedIn()) {  
 die(json\_encode(['success' => false, 'error' => 'Not authorized']));  
}  
try {  
 // Валидация входных данных  
 $modelFile = $\_POST['model'] ?? '';  
 if (empty($modelFile)) {  
 throw new Exception("Не указан файл модели");  
 }  
 $modelPath = *MODEL\_UPLOAD\_DIR* . $modelFile;  
 if (!file\_exists($modelPath)) {  
 throw new Exception("Файл модели не найден");  
 }  
 // Получение параметров слайсинга  
 $settings = [  
 'material' => $\_POST['material'] ?? 'PLA',  
 'nozzle\_size' => (float)($\_POST['nozzle\_size'] ?? 0.4),  
 'layer\_height' => (float)($\_POST['layer\_height'] ?? 0.2),  
 'infill\_density' => (int)($\_POST['infill\_density'] ?? 20),  
 'support' => isset($\_POST['generate\_support']) ? 'true' : 'false',  
 'brim' => isset($\_POST['add\_brim']) ? 'brim' : 'none'  
 ];  
 // Генерация имени выходного файла  
 $outputFile = 'slice\_'.time().'\_'.bin2hex(random\_bytes(4)).'.gcode';  
 $gcodePath = *GCODE\_UPLOAD\_DIR* . $outputFile;  
 // Формирование команды для CuraEngine  
 $command = sprintf(  
 '"%s" slice -v -j "%s" -l "%s" -o "%s" '.  
 '-s material=%s '.  
 '-s nozzle\_size=%s '.  
 '-s layer\_height=%s '.  
 '-s infill\_sparse\_density=%d '.  
 '-s support\_enable=%s '.  
 '-s adhesion\_type=%s',  
 *CURA\_ENGINE\_PATH*,  
 *PRINTER\_PROFILE\_PATH*,  
 escapeshellarg($modelPath),  
 escapeshellarg($gcodePath),  
 $settings['material'],  
 $settings['nozzle\_size'],  
 $settings['layer\_height'],  
 $settings['infill\_density'],  
 $settings['support'],  
 $settings['brim']  
 );  
 // Выполнение команды  
 exec($command, $output, $returnCode);  
 if ($returnCode !== 0 || !file\_exists($gcodePath)) {  
 throw new Exception("Ошибка слайсинга: ".implode("\n", $output));  
 }  
 // Возвращаем результат  
 echo json\_encode([  
 'success' => true,  
 'filename' => $outputFile,  
 'gcode' => file\_get\_contents($gcodePath),  
 'print\_time' => 'N/A',  
 'filament\_used' => 'N/A'  
 ]);  
} catch (Exception $e) {  
 error\_log('Slice error: '.$e->getMessage());  
 echo json\_encode([  
 'success' => false,  
 'error' => $e->getMessage()  
 ]);  
}

SlicerProfile.php

<?php  
require\_once *\_\_DIR\_\_*.'/Config.php';  
class SlicerProfile {  
 private $printerName;  
 public function \_\_construct($printerName = 'Anycubic Kobra 3') {  
 if (!isset(*PRINTER\_PROFILES*[$printerName])) {  
 throw new Exception("Printer profile not found");  
 }  
 $this->printerName = $printerName;  
 }  
 public function getStartGcode($material) {  
 $profile = *PRINTER\_PROFILES*[$this->printerName];  
 $materialSettings = *MATERIALS*[$material];  
 $gcode = [];  
 $gcode[] = "; Start G-code for {$this->printerName}";  
 $gcode[] = "M140 S{$materialSettings['bed\_temp'][1]} ; Set bed temp";  
 $gcode[] = "M104 S{$materialSettings['extruder\_temp'][1]} ; Set extruder temp";  
 $gcode[] = "G28 ; Home all axes";  
 $gcode[] = "G29 ; Auto bed leveling";  
 $gcode[] = "M190 S{$materialSettings['bed\_temp'][1]} ; Wait for bed temp";  
 $gcode[] = "M109 S{$materialSettings['extruder\_temp'][1]} ; Wait for extruder temp";  
 $gcode[] = "G21 ; Set units to millimeters";  
 $gcode[] = "G90 ; Use absolute positioning";  
 $gcode[] = "M82 ; Set extruder to absolute mode";  
 return implode("\n", $gcode);  
 }  
 public function getEndGcode() {  
 $gcode = [];  
 $gcode[] = "; End G-code for {$this->printerName}";  
 $gcode[] = "G91 ; Relative positioning";  
 $gcode[] = "G1 Z10 F900 ; Lift nozzle";  
 $gcode[] = "G90 ; Absolute positioning";  
 $gcode[] = "G1 X0 Y220 F9000 ; Present print";  
 $gcode[] = "M104 S0 ; Turn off extruder";  
 $gcode[] = "M140 S0 ; Turn off bed";  
 $gcode[] = "M107 ; Turn off fan";  
 $gcode[] = "M84 ; Disable steppers";  
  
 return implode("\n", $gcode);  
 }  
 public function getNozzleSizes() {  
 return *PRINTER\_PROFILES*[$this->printerName]['nozzle\_sizes'];  
 }  
}

D:\DIPLOMKA\3d-printer-slicer\templates

dashboard.php

<?php  
// Убедимся, что пути правильные относительно расположения dashboard.php  
require\_once *\_\_DIR\_\_*.'/../includes/Auth.php';  
require\_once *\_\_DIR\_\_*.'/../includes/Database.php';  
require\_once *\_\_DIR\_\_*.'/../includes/Config.php';  
require\_once *\_\_DIR\_\_*.'/../includes/FileUploader.php';  
// Инициализация  
$auth = new Auth();  
if (!$auth->isLoggedIn()) {  
 header("Location: login.php");  
 exit;  
}  
$user = $auth->getCurrentUser();  
$db = new Database();  
// Обработка загрузки файла  
$uploadError = '';  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_FILES['model'])) {  
 try {  
 $filename = FileUploader::*uploadModel*($\_FILES['model']);  
 header("Location: slicer.php?model=" . urlencode($filename));  
 exit;  
 } catch (Exception $e) {  
 $uploadError = $e->getMessage();  
 }  
}  
// Получаем историю печати  
$prints = $db->query(  
 "SELECT ph.\*, u.username   
 FROM print\_history ph  
 JOIN users u ON ph.user\_id = u.id  
 WHERE ph.user\_id = ?   
 ORDER BY ph.created\_at DESC   
 LIMIT 5",  
 [$user['id']]  
);  
);  
?>  
<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>LiteSlicer - Dashboard</title>  
 <style>  
 :root {  
 --main-bg: #1a1a2e;  
 --card-bg: #16213e;  
 --accent-blue: #0f3460;  
 --accent-red: #e94560;  
 --text-light: #ffffff;  
 --text-dim: #b8b8b8;  
 }  
 body {  
 background-color: var(--main-bg);  
 color: var(--text-light);  
 font-family: 'Segoe UI', sans-serif;  
 margin: 0;  
 padding: 20px;  
 }  
 .container {  
 max-width: 1200px;  
 margin: 0 auto;  
 }  
 .card {  
 background: var(--card-bg);  
 border-radius: 8px;  
 padding: 20px;  
 margin-bottom: 20px;  
 box-shadow: 0 4px 6px rgba(0,0,0,0.1);  
 }  
 .card-header {  
 border-bottom: 2px solid var(--accent-red);  
 padding-bottom: 10px;  
 margin-bottom: 15px;  
 color: var(--text-light);  
 }  
 .btn {  
 background: var(--accent-red);  
 color: white;  
 border: none;  
 padding: 10px 15px;  
 border-radius: 4px;  
 cursor: pointer;  
 font-weight: bold;  
 }  
 .btn:hover {  
 background: #d43d57;  
 }  
 .file-upload {  
 border: 2px dashed var(--accent-blue);  
 padding: 30px;  
 text-align: center;  
 margin: 20px 0;  
 border-radius: 8px;  
 }  
 </style>  
</head>  
<body>  
<div class="container">  
 <h1>LiteSlicer</h1>  
 <div class="card">  
 <div class="card-header">  
 <h2>Загрузка модели</h2>  
 </div>  
 <?php if ($uploadError): ?>  
 <div style="color: var(--accent-red); margin-bottom: 15px;">  
 Ошибка: <?= htmlspecialchars($uploadError) ?>  
 </div>  
 <?php endif; ?>  
 <form method="post" enctype="multipart/form-data">  
 <div class="file-upload">  
 <input type="file" name="model" accept=".stl,.obj" required>  
 <p>Поддерживаемые форматы: STL, OBJ (макс. 50MB)</p>  
 </div>  
 <button type="submit" class="btn">ПОДГРУЗИТЬ И НАРЕЗАТЬ</button>  
 </form>  
 </div>  
 <div class="card">  
 <div class="card-header">  
 <h2>История печати</h2>  
 </div>  
 <?php if (empty($prints)): ?>  
 <p>Нет данных о предыдущих печатях</p>  
 <?php else: ?>  
 <div style="display: grid; grid-template-columns: repeat(auto-fill, minmax(250px, 1fr)); gap: 15px;">  
 <?php foreach ($prints as $print): ?>  
 <div style="background: rgba(15, 52, 96, 0.3); padding: 10px; border-radius: 6px;">  
 <h3><?= htmlspecialchars($print['model\_name']) ?></h3>  
 <p>Материал: <?= htmlspecialchars($print['material']) ?></p>  
 <p>Дата: <?= date('d.m.Y', strtotime($print['created\_at'])) ?></p>  
 </div>  
 <?php endforeach; ?>  
 </div>  
 <?php endif; ?>  
 </div>  
</div>  
</body>  
</html>

footer.php

</main>  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>  
<script>  
 // Закрытие модальных окон после успешных действий  
 <?php if ($error && isset($\_POST['register']) && strpos($error, 'success') !== false): ?>  
 *document*.addEventListener('DOMContentLoaded', function() {  
 var registerModal = bootstrap.Modal.*getInstance*(*document*.getElementById('registerModal'));  
 registerModal.hide();  
 *document*.getElementById('reset-view').addEventListener('click', () => {  
 if (*model*) {  
 *controls*.reset();  
 *camera*.position.z = 5;  
 *camera*.position.y = 2.5;  
 *camera*.lookAt(0, 0, 0);  
 *controls*.target.set(0, 0, 0);  
 *controls*.update();  
 }  
 });  
 var loginModal = new bootstrap.Modal(*document*.getElementById('loginModal'));  
 loginModal.show();  
 });  
 <?php endif; ?>  
</script>  
</body>  
</html>

header.php

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>3D Printer Lite Slicer</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.8.1/font/bootstrap-icons.css" rel="stylesheet">  
 <!-- Three.js и дополнительные модули -->  
 <script src="https://cdn.jsdelivr.net/npm/three@0.132.2/build/three.min.js"></script>  
 <script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/controls/OrbitControls.js"></script>  
 <script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/loaders/STLLoader.js"></script>  
 <script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/loaders/OBJLoader.js"></script>  
 <!-- Библиотека для слайсинга (используем CuraEngine WASM) -->  
 <script src="https://cdn.jsdelivr.net/npm/cura-wasm@1.0.0/dist/cura-wasm.js"></script>  
  
 <!-- Дополнительные утилиты -->  
 <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>  
 <!DOCTYPE html>  
 <html lang="en">  
 <head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>NEON SLICER 3000</title>  
 <!-- Киберпанк шрифты -->  
 <link href="https://fonts.googleapis.com/css2?family=Orbitron:wght@400;700&family=Rajdhani:wght@500;700&display=swap" rel="stylesheet">  
 <!-- Стили -->  
 <style>  
 :root {  
 --neon-red: #00ff0d;  
 --neon-yellow: #f5d742;  
 --neon-blue: #0abdc6;  
 --neon-purple: #d300c5;  
 --dark-bg: #0c0c1a;  
 }  
 body {  
 background-color: var(--dark-bg);  
 color: #e0e0e0;  
 font-family: 'Rajdhani', sans-serif;  
 overflow-x: hidden;  
 }  
 .cyber-card {  
 background: rgba(12, 12, 26, 0.8);  
 border: 1px solid var(--neon-purple);  
 border-radius: 0;  
 box-shadow: 0 0 15px var(--neon-purple);  
 margin-bottom: 25px;  
 transition: all 0.3s;  
 }  
 .cyber-card:hover {  
 box-shadow: 0 0 25px var(--neon-blue);  
 border-color: var(--neon-blue);  
 }  
 .cyber-header {  
 background: linear-gradient(90deg, var(--neon-red), var(--neon-purple));  
 color: black;  
 font-family: 'Orbitron', sans-serif;  
 font-weight: 700;  
 letter-spacing: 2px;  
 border-bottom: 2px solid var(--neon-yellow);  
 }  
 .cyber-btn {  
 background: transparent;  
 color: var(--neon-yellow);  
 border: 1px solid var(--neon-yellow);  
 font-family: 'Orbitron', sans-serif;  
 letter-spacing: 1px;  
 transition: all 0.3s;  
 position: relative;  
 overflow: hidden;  
 }  
 #viewer-container {  
 position: relative;  
 width: 100%;  
 height: 600px;  
 background: #111;  
 border: 1px solid #0abdc6;  
 overflow: hidden;  
 }  
 #viewer {  
 width: 100%;  
 height: 100%;  
 }  
 .control-panel {  
 position: absolute;  
 bottom: 20px;  
 right: 20px;  
 z-index: 100;  
 display: flex;  
 gap: 10px;  
 }  
 .control-btn {  
 background: rgba(0, 0, 0, 0.7);  
 border: 1px solid #0abdc6;  
 color: #0abdc6;  
 width: 40px;  
 height: 40px;  
 border-radius: 50%;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 cursor: pointer;  
 font-size: 20px;  
 }  
 .control-btn:hover {  
 background: #0abdc6;  
 color: #000;  
 }  
 .cyber-btn:hover {  
 color: black;  
 background: var(--neon-yellow);  
 box-shadow: 0 0 15px var(--neon-yellow);  
 }  
 /\* Эффекты неона \*/  
 .neon-effect {  
 text-shadow: 0 0 5px currentColor;  
 }  
 /\* Глитч эффекты \*/  
 @keyframes glitch {  
 0% { transform: translate(0); }  
 20% { transform: translate(-2px, 2px); }  
 40% { transform: translate(-2px, -2px); }  
 60% { transform: translate(2px, 2px); }  
 80% { transform: translate(2px, -2px); }  
 100% { transform: translate(0); }  
 }  
 .glitch-effect:hover {  
 animation: glitch 0.5s linear infinite;  
 }  
 <!-- Хакерская матрица фона -->  
 <div id="matrix-effect" style="position:fixed;top:0;left:0;z-index:-1;opacity:0.1"></div>  
 #viewer-container {  
 position: relative;  
 width: 100%;  
 height: 600px;  
 background-color: #f0f0f0;  
 border-radius: 5px;  
 overflow: hidden;  
 }  
 #viewer {  
 width: 100%;  
 height: 100%;  
 }  
 .form-range::-webkit-slider-thumb {  
 background: #0d6efd;  
 }  
 .btn-zoom {  
 margin-right: 5px;  
 }  
 body { padding-top: 20px; }  
 .modal-backdrop { background-color: rgba(0,0,0,0.5); }  
 </style>  
 <style>  
 #viewer-container {  
 position: relative;  
 width: 100%;  
 height: 600px;  
 border: 1px solid #ddd;  
 }  
 .form-range::-webkit-slider-thumb {  
 background: #0d6efd;  
 }  
 #gcode-preview {  
 height: 200px;  
 overflow-y: auto;  
 background: #f8f9fa;  
 font-family: monospace;  
 white-space: pre;  
 padding: 10px;  
 }  
 </style>  
</head>  
<body>

home.php

<div class="container text-center mt-5">  
 <div class="row">  
 <div class="col-lg-8 mx-auto">  
 <h1 class="display-4 mb-4">3D Printer Web Slicer</h1>  
 <p class="lead">Professional slicing tool for Anycubic Kobra 3</p>  
 <div class="mt-5">  
 <button class="btn btn-primary btn-lg mx-2" data-bs-toggle="modal" data-bs-target="#loginModal">  
 <i class="bi bi-box-arrow-in-right"></i> Login  
 </button>  
 <button class="btn btn-success btn-lg mx-2" data-bs-toggle="modal" data-bs-target="#registerModal">  
 <i class="bi bi-person-plus"></i> Register  
 </button>  
 </div>  
 </div>  
 </div>  
</div>  
<!-- Модальное окно входа -->  
<div class="modal fade" id="loginModal" tabindex="-1" aria-hidden="true">  
 <div class="modal-dialog modal-dialog-centered">  
 <div class="modal-content">  
 <form method="POST" action="">  
 <div class="modal-header">  
 <h5 class="modal-title">Login to your account</h5>  
 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>  
 </div>  
 <div class="modal-body">  
 <?php if ($error && isset($\_POST['login'])): ?>  
 <div class="alert alert-danger"><?= htmlspecialchars($error) ?></div>  
 <?php endif; ?>  
 <div class="mb-3">  
 <label for="loginEmail" class="form-label">Email address</label>  
 <input type="email" class="form-control" id="loginEmail" name="email" required>  
 </div>  
 <div class="mb-3">  
 <label for="loginPassword" class="form-label">Password</label>  
 <input type="password" class="form-control" id="loginPassword" name="password" required>  
 </div>  
 <div class="d-grid gap-2">  
 <button type="submit" name="login" class="btn btn-primary">Login</button>  
 </div>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>  
<!-- Модальное окно регистрации -->  
<div class="modal fade" id="registerModal" tabindex="-1" aria-hidden="true">  
 <div class="modal-dialog modal-dialog-centered">  
 <div class="modal-content">  
 <form method="POST" action="">  
 <div class="modal-header">  
 <h5 class="modal-title">Create new account</h5>  
 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>  
 </div>  
 <div class="modal-body">  
 <?php if ($error && isset($\_POST['register'])): ?>  
 <div class="alert <?= strpos($error, 'success') !== false ? 'alert-success' : 'alert-danger' ?>">  
 <?= htmlspecialchars($error) ?>  
 </div>  
 <?php endif; ?>  
 <div class="mb-3">  
 <label for="registerUsername" class="form-label">Username</label>  
 <input type="text" class="form-control" id="registerUsername" name="username" required>  
 </div>  
 <div class="mb-3">  
 <label for="registerEmail" class="form-label">Email address</label>  
 <input type="email" class="form-control" id="registerEmail" name="email" required>  
 </div>  
 <div class="mb-3">  
 <label for="registerPassword" class="form-label">Password</label>  
 <input type="password" class="form-control" id="registerPassword" name="password" required>  
 </div>  
 <div class="mb-3">  
 <label for="registerConfirm" class="form-label">Confirm Password</label>  
 <input type="password" class="form-control" id="registerConfirm" name="confirm\_password" required>  
 </div>  
 <div class="d-grid gap-2">  
 <button type="submit" name="register" class="btn btn-success">Register</button>  
 </div>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>

slicer.php

<?php  
require\_once *\_\_DIR\_\_*.'/../includes/Config.php';  
require\_once *\_\_DIR\_\_*.'/../includes/Auth.php';  
$auth = new Auth();  
if (!$auth->isLoggedIn()) {  
 header('HTTP/1.1 403 Forbidden');  
 die(json\_encode(['success' => false, 'error' => 'Not authorized']));  
}  
$modelFile = $\_GET['model'] ?? '';  
$modelPath = *MODEL\_UPLOAD\_DIR* . $modelFile;  
if (!file\_exists($modelPath)) {  
 header("Location: dashboard.php");  
 exit;  
}  
?>  
<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>NEON SLICER 3000 | 3D Control</title>  
 <style>  
 :root {  
 --neon-red: #07ff8f;  
 --neon-blue: #0abdc6;  
 --neon-purple: #0004d3;  
 --neon-yellow: #f5d742;  
 --dark-bg: #0c0c1a;  
 }  
 body {  
 margin: 0;  
 background-color: var(--dark-bg);  
 color: #e0e0e0;  
 font-family: 'Rajdhani', sans-serif;  
 overflow: hidden;  
 }  
 #viewer-container {  
 position: relative;  
 width: 100%;  
 height: 70vh;  
 border: 1px solid var(--neon-blue);  
 }  
 #controls-overlay {  
 position: absolute;  
 top: 20px;  
 right: 20px;  
 background: rgba(20, 20, 40, 0.9);  
 border: 1px solid var(--neon-purple);  
 padding: 15px;  
 z-index: 1000;  
 width: 280px;  
 }  
 .control-group {  
 margin-bottom: 15px;  
 }  
 .control-title {  
 color: var(--neon-yellow);  
 margin-bottom: 8px;  
 font-size: 0.9rem;  
 }  
 .axis-controls {  
 display: flex;  
 flex-wrap: wrap;  
 gap: 5px;  
 }  
 .control-btn {  
 background: rgba(0,0,0,0.7);  
 border: 1px solid var(--neon-blue);  
 color: var(--neon-blue);  
 padding: 8px 12px;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
 .control-btn:hover {  
 background: var(--neon-blue);  
 color: #000;  
 }  
 </style>  
</head>  
<body>  
<div id="viewer-container">  
 <div id="viewer"></div>  
 <!-- Панель управления моделью -->  
 <div id="controls-overlay">  
 <div class="control-group">  
 <div class="control-title">ПЕРЕМЕЩЕНИЕ</div>  
 <div class="axis-controls">  
 <button class="control-btn" data-axis="x" data-direction="+">X+</button>  
 <button class="control-btn" data-axis="x" data-direction="-">X-</button>  
 <button class="control-btn" data-axis="y" data-direction="+">Y+</button>  
 <button class="control-btn" data-axis="y" data-direction="-">Y-</button>  
 <button class="control-btn" data-axis="z" data-direction="+">Z+</button>  
 <button class="control-btn" data-axis="z" data-direction="-">Z-</button>  
 </div>  
 </div>  
 <div class="control-group">  
 <div class="control-title">ВРАЩЕНИЕ</div>  
 <div class="axis-controls">  
 <button class="control-btn" data-rotate="x">RX</button>  
 <button class="control-btn" data-rotate="y">RY</button>  
 <button class="control-btn" data-rotate="z">RZ</button>  
 </div>  
 </div>  
 <div class="control-group">  
 <div class="control-title">МАСШТАБ</div>  
 <div class="axis-controls">  
 <button class="control-btn" id="scale-up">+</button>  
 <button class="control-btn" id="scale-down">-</button>  
 <button class="control-btn" id="reset-scale">Сброс</button>  
 </div>  
 </div>  
 <div class="control-group">  
 <div class="control-title">КООРДИНАТЫ</div>  
 <div id="position-info" style="font-family: monospace;">  
 X: 0.0 | Y: 0.0 | Z: 0.0  
 </div>  
 </div>  
 </div>  
</div>  
<!-- Three.js и компоненты -->  
<script src="https://cdn.jsdelivr.net/npm/three@0.132.2/build/three.min.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/controls/OrbitControls.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/loaders/STLLoader.js"></script>  
<script>  
 // Инициализация сцены  
 const *scene* = new THREE.Scene();  
 *scene*.background = new THREE.Color(0x111126);  
 // Камера  
 const *camera* = new THREE.PerspectiveCamera(75, *window*.innerWidth / *window*.innerHeight, 0.1, 1000);  
 *camera*.position.set(0, 150, 300);  
 // Рендерер  
 const *renderer* = new THREE.WebGLRenderer({ antialias: true });  
 *renderer*.setSize(*window*.innerWidth, *window*.innerHeight);  
 *document*.getElementById('viewer').appendChild(*renderer*.domElement);  
 // Освещение  
 const *ambientLight* = new THREE.AmbientLight(0x404040);  
 *scene*.add(*ambientLight*);  
 const *directionalLight* = new THREE.DirectionalLight(0xffffff, 0.8);  
 *directionalLight*.position.set(1, 1, 1);  
 *scene*.add(*directionalLight*);  
 // Платформа  
 const *platformSize* = 220;  
 const *platformGeometry* = new THREE.BoxGeometry(*platformSize*/10, 2, *platformSize*/10);  
 const *platformMaterial* = new THREE.MeshPhongMaterial({  
 color: 0x333333,  
 transparent: true,  
 opacity: 0.7  
 });  
 const *platform* = new THREE.Mesh(*platformGeometry*, *platformMaterial*);  
 *platform*.position.y = -1;  
 *scene*.add(*platform*);  
 // Загрузка модели  
 let *model* = null;  
 const *loader* = new THREE.STLLoader();  
 function loadModel(url) {  
 *loader*.load(url, function(geometry) {  
 if (*model*) *scene*.remove(*model*);  
 geometry.computeBoundingBox();  
 const bbox = geometry.boundingBox;  
 const center = new THREE.Vector3();  
 bbox.getCenter(center);  
 const material = new THREE.MeshPhongMaterial({  
 color: 0x0abdc6,  
 specular: 0x111111,  
 shininess: 30  
 });  
 *model* = new THREE.Mesh(geometry, material);  
 *model*.position.sub(center);  
 const maxDim = *Math*.max(  
 bbox.max.x - bbox.min.x,  
 bbox.max.y - bbox.min.y,  
 bbox.max.z - bbox.min.z  
 );  
 const scale = (*platformSize*/10) \* 0.8 / maxDim;  
 *model*.scale.set(scale, scale, scale);  
 *model*.position.y = (bbox.max.y - bbox.min.y) \* scale / 2;  
 *scene*.add(*model*);  
 updatePositionInfo();  
 });  
 }  
 // Загружаем модель из URL  
 loadModel('/uploads/models/<?= $modelFile ?>');  
 // Управление моделью  
 const *MOVE\_STEP* = 5;  
 const *ROTATE\_STEP* = *Math*.PI / 8;  
 const *SCALE\_STEP* = 0.1;  
 *document*.querySelectorAll('[data-axis]').forEach(btn => {  
 btn.addEventListener('click', () => {  
 if (!*model*) return;  
 const axis = btn.dataset.axis;  
 const dir = btn.dataset.direction === '+' ? 1 : -1;  
 *model*.position[axis] += *MOVE\_STEP* \* dir;  
 constrainModel();  
 updatePositionInfo();  
 });  
 });  
 *document*.querySelectorAll('[data-rotate]').forEach(btn => {  
 btn.addEventListener('click', () => {  
 if (!*model*) return;  
 *model*.rotation[btn.dataset.rotate] += *ROTATE\_STEP*;  
 updatePositionInfo();  
 });  
 });  
 *document*.getElementById('scale-up').addEventListener('click', () => {  
 if (!*model*) return;  
 *model*.scale.multiplyScalar(1 + *SCALE\_STEP*);  
 updatePositionInfo();  
 });  
 *document*.getElementById('scale-down').addEventListener('click', () => {  
 if (!*model*) return;  
 *model*.scale.multiplyScalar(1 - *SCALE\_STEP*);  
 updatePositionInfo();  
 });  
 *document*.getElementById('reset-scale').addEventListener('click', () => {  
 if (!*model*) return;  
 *model*.scale.set(1, 1, 1);  
 updatePositionInfo();  
 });  
 function constrainModel() {  
 // Ограничение перемещения по платформе  
 const halfPlatform = *platformSize* / 20;  
 *model*.position.x = *Math*.max(-halfPlatform, *Math*.min(halfPlatform, *model*.position.x));  
 *model*.position.z = *Math*.max(-halfPlatform, *Math*.min(halfPlatform, *model*.position.z));  
 }  
 function updatePositionInfo() {  
 if (!*model*) return;  
 *document*.getElementById('position-info').innerHTML = `  
 X: ${*model*.position.x.toFixed(1)} |  
 Y: ${*model*.position.y.toFixed(1)} |  
 Z: ${*model*.position.z.toFixed(1)} `;  
 }  
 // Анимация  
 function animate() {  
 requestAnimationFrame(animate);  
 *renderer*.render(*scene*, *camera*);  
 }  
 animate();  
 // Реакция на изменение размера  
 *window*.addEventListener('resize', () => {  
 *camera*.aspect = *window*.innerWidth / *window*.innerHeight;  
 *camera*.updateProjectionMatrix();  
 *renderer*.setSize(*window*.innerWidth, *window*.innerHeight);  
 });  
</script>  
</body>  
</html>

D:\DIPLOMKA\3d-printer-slicer\uploads

.htaccess

php\_value upload\_max\_filesize 200M  
php\_value post\_max\_size 210M  
php\_value memory\_limit 256M  
php\_value max\_execution\_time 300  
php\_value max\_input\_time 300

composer.json

{

"name": "vendor\_name/3d-printer-slicer",

"description": "description",

"minimum-stability": "stable",

"license": "proprietary",

"authors": [

{

"name": "захар",

"email": "email@example.com"

}

],

"require": {

}

}

dashboard.php

<?php  
require\_once *\_\_DIR\_\_*.'/includes/Auth.php';  
require\_once *\_\_DIR\_\_*.'/includes/Database.php';  
require\_once *\_\_DIR\_\_*.'/includes/Config.php';  
$auth = new Auth();  
if (!$auth->isLoggedIn()) {  
 header("Location: login.php");  
 exit;  
}  
$user = $auth->getCurrentUser();  
$db = new Database();  
// Get print history  
$prints = $db->query(  
 "SELECT \* FROM prints WHERE user\_id = ? ORDER BY created\_at DESC LIMIT 5",  
 [$user['id']]  
);  
// Handle file upload  
$uploadError = '';  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_FILES['model'])) {  
 try {  
 require\_once *\_\_DIR\_\_*.'/includes/FileUploader.php';  
 $filename = FileUploader::*uploadModel*($\_FILES['model']);  
 header("Location: slicer.php?model=" . urlencode($filename));  
 exit;  
 } catch (Exception $e) {  
 $uploadError = $e->getMessage();  
 }  
}  
// Handle platform size save  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['save\_platform\_size'])) {  
 $platformWidth = (int)$\_POST['platform\_width'];  
 $platformDepth = (int)$\_POST['platform\_depth'];  
 $platformHeight = (int)$\_POST['platform\_height'];  
 // Validate dimensions  
 if ($platformWidth >= 100 && $platformWidth <= 500 &&  
 $platformDepth >= 100 && $platformDepth <= 500 &&  
 $platformHeight >= 100 && $platformHeight <= 500) {  
 $\_SESSION['platform\_width'] = $platformWidth;  
 $\_SESSION['platform\_depth'] = $platformDepth;  
 $\_SESSION['platform\_height'] = $platformHeight;  
 }  
}  
// Get current platform size from session or use defaults  
$currentPlatformWidth = $\_SESSION['platform\_width'] ?? 220;  
$currentPlatformDepth = $\_SESSION['platform\_depth'] ?? 220;  
$currentPlatformHeight = $\_SESSION['platform\_height'] ?? 250;  
?>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Lite Slicer - Dashboard</title>  
 <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap" rel="stylesheet">  
 <style>  
 .print-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 15px;  
 }  
 .print-table th {  
 color: var(--primary-blue);  
 font-weight: 500;  
 text-align: left;  
 padding: 12px;  
 border-bottom: 1px solid var(--primary-blue);  
 }  
 .print-table td {  
 padding: 12px;  
 border-bottom: 1px solid rgba(100, 181, 246, 0.3);  
 }  
 .print-table tr:hover {  
 background: rgba(10, 189, 198, 0.1);  
 }  
 .table-responsive {  
 overflow-x: auto;  
 }  
 :root {  
 --primary-blue: #0abdc6;  
 --dark-blue: #1a237e;  
 --light-blue: #e3f2fd;  
 --accent-blue: #64b5f6;  
 --text-dark: #212121;  
 --text-light: #ffffff;  
 --card-bg: rgba(26, 35, 126, 0.7);  
 }  
 body {  
 margin: 0;  
 padding: 0;  
 font-family: 'Roboto', sans-serif;  
 background-color: #0c0c1a;  
 color: var(--text-light);  
 min-height: 100vh;  
 }  
 #matrix-effect {  
 position: fixed;  
 top: 0;  
 left: 0;  
 z-index: -1;  
 opacity: 0.1;  
 width: 100%;  
 height: 100%;  
 }  
 .container {  
 max-width: 1200px;  
 margin: 0 auto;  
 padding: 20px;  
 }  
 header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 30px;  
 padding-bottom: 20px;  
 border-bottom: 1px solid rgba(100, 181, 246, 0.3);  
 }  
 .logo {  
 display: flex;  
 align-items: center;  
 }  
 .logo img {  
 height: 40px;  
 margin-right: 10px;  
 }  
 .logo-text {  
 font-size: 1.5rem;  
 font-weight: 700;  
 background: linear-gradient(90deg, #64b5f6, #0abdc6);  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 }  
 .user-menu {  
 display: flex;  
 align-items: center;  
 }  
 .user-avatar {  
 width: 40px;  
 height: 40px;  
 border-radius: 50%;  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-weight: 700;  
 margin-right: 10px;  
 }  
 .logout-btn {  
 background: transparent;  
 border: 1px solid var(--primary-blue);  
 color: var(--primary-blue);  
 padding: 8px 15px;  
 border-radius: 5px;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
 .logout-btn:hover {  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 }  
 .dashboard-grid {  
 display: grid;  
 grid-template-columns: 300px 1fr;  
 gap: 20px;  
 }  
 .sidebar {  
 background: var(--card-bg);  
 border: 1px solid var(--primary-blue);  
 border-radius: 10px;  
 padding: 20px;  
 backdrop-filter: blur(5px);  
 }  
 .sidebar-title {  
 color: var(--primary-blue);  
 font-size: 1.2rem;  
 margin-bottom: 20px;  
 padding-bottom: 10px;  
 border-bottom: 1px solid rgba(100, 181, 246, 0.3);  
 }  
 .printer-info {  
 margin-bottom: 30px;  
 }  
 .info-item {  
 margin-bottom: 15px;  
 }  
 .info-label {  
 color: var(--accent-blue);  
 font-size: 0.9rem;  
 margin-bottom: 5px;  
 }  
 .info-value {  
 font-size: 1rem;  
 }  
 .main-content {  
 display: flex;  
 flex-direction: column;  
 gap: 20px;  
 }  
 .card {  
 background: var(--card-bg);  
 border: 1px solid var(--primary-blue);  
 border-radius: 10px;  
 padding: 20px;  
 backdrop-filter: blur(5px);  
 }  
 .card-title {  
 color: var(--primary-blue);  
 font-size: 1.2rem;  
 margin-bottom: 20px;  
 padding-bottom: 10px;  
 border-bottom: 1px solid rgba(100, 181, 246, 0.3);  
 }  
 .upload-area {  
 border: 2px dashed var(--primary-blue);  
 border-radius: 10px;  
 padding: 30px;  
 text-align: center;  
 margin-bottom: 20px;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
 .upload-area:hover {  
 background: rgba(10, 189, 198, 0.1);  
 }  
 .upload-icon {  
 font-size: 3rem;  
 color: var(--primary-blue);  
 margin-bottom: 15px;  
 }  
 .upload-text {  
 margin-bottom: 10px;  
 }  
 .file-input {  
 display: none;  
 }  
 .btn {  
 background: linear-gradient(90deg, var(--primary-blue), #64b5f6);  
 color: var(--text-dark);  
 border: none;  
 padding: 12px;  
 border-radius: 5px;  
 font-weight: 700;  
 cursor: pointer;  
 width: 100%;  
 transition: all 0.3s;  
 }  
 .btn:hover {  
 box-shadow: 0 0 15px rgba(10, 189, 198, 0.5);  
 }  
 .error-message {  
 color: #ff5252;  
 margin-bottom: 20px;  
 text-align: center;  
 }  
 .prints-grid {  
 display: grid;  
 grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));  
 gap: 15px;  
 }  
 .print-card {  
 background: rgba(10, 189, 198, 0.1);  
 border: 1px solid var(--primary-blue);  
 border-radius: 8px;  
 padding: 15px;  
 transition: all 0.3s;  
 }  
 .print-card:hover {  
 transform: translateY(-5px);  
 box-shadow: 0 10px 20px rgba(10, 189, 198, 0.2);  
 }  
 .print-name {  
 font-weight: 500;  
 margin-bottom: 10px;  
 color: var(--primary-blue);  
 }  
 .print-meta {  
 display: flex;  
 justify-content: space-between;  
 font-size: 0.9rem;  
 color: var(--accent-blue);  
 }  
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: var(--accent-blue);  
 }  
 /\* Platform settings styles \*/  
 .platform-settings {  
 margin-top: 20px;  
 padding: 15px;  
 background: rgba(10, 189, 198, 0.1);  
 border-radius: 5px;  
 border: 1px solid var(--primary-blue);  
 }  
 .platform-settings-title {  
 color: var(--primary-blue);  
 margin-bottom: 10px;  
 font-weight: 500;  
 }  
 .platform-form {  
 display: grid;  
 grid-template-columns: repeat(3, 1fr);  
 gap: 10px;  
 align-items: center;  
 }  
 .platform-form-group {  
 display: flex;  
 flex-direction: column;  
 }  
 .platform-form-label {  
 color: var(--accent-blue);  
 font-size: 0.8rem;  
 margin-bottom: 5px;  
 }  
 .platform-form input {  
 width: 100%;  
 padding: 8px;  
 border: 1px solid var(--primary-blue);  
 border-radius: 4px;  
 background: rgba(0,0,0,0.3);  
 color: var(--text-light);  
 }  
 .platform-form .btn {  
 grid-column: span 3;  
 padding: 8px 12px;  
 width: 100%;  
 }  
 </style>  
</head>  
<body>  
<div id="matrix-effect"></div>  
<div class="container">  
 <header>  
 <div class="logo">  
 <img src="photo/photo1.jpg" alt="Lite Slicer Logo">  
 <div class="logo-text">Lite Slicer</div>  
 </div>  
 <div class="user-menu">  
 <div class="user-avatar"><?= strtoupper(substr($user['username'], 0, 1)) ?></div>  
 <a href="logout.php" class="logout-btn">Logout</a>  
 </div>  
 </header>  
 <div class="dashboard-grid">  
 <div class="sidebar">  
 <h3 class="sidebar-title">PRINTER STATUS</h3>  
 <div class="printer-info">  
 <div class="info-item">  
 <div class="info-label">Model</div>  
 <div class="info-value">Anycubic Kobra 3</div>  
 </div>  
 <div class="info-item">  
 <div class="info-label">Status</div>  
 <div class="info-value" style="color: #69f0ae;">Online</div>  
 </div>  
 <div class="info-item">  
 <div class="info-label">Build Volume</div>  
 <div class="info-value"><?= $currentPlatformWidth ?>x<?= $currentPlatformDepth ?>x<?= $currentPlatformHeight ?>mm</div>  
 </div>  
 <div class="info-item">  
 <div class="info-label">Nozzle</div>  
 <div class="info-value">0.4mm</div>  
 </div>  
 </div>  
 </div>  
 <div class="main-content">  
 <div class="card">  
 <h3 class="card-title">UPLOAD MODEL</h3>  
 <?php if ($uploadError): ?>  
 <div class="error-message">Error: <?= htmlspecialchars($uploadError) ?></div>  
 <?php endif; ?>  
 <form method="post" enctype="multipart/form-data">  
 <div class="upload-area" onclick="*document*.getElementById('model-file').click()">  
 <div class="upload-icon">  
 <svg xmlns="http://www.w3.org/2000/svg" width="48" height="48" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">  
 <path d="M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4"></path>  
 <polyline points="17 8 12 3 7 8"></polyline>  
 <line x1="12" y1="3" x2="12" y2="15"></line>  
 </svg>  
 </div>  
 <div class="upload-text">Click to upload or drag and drop</div>  
 <div style="color: var(--accent-blue); font-size: 0.9rem;">STL or OBJ files (max 50MB)</div>  
 <input type="file" id="model-file" name="model" class="file-input" accept=".stl,.obj" required>  
 </div>  
 <button type="submit" class="btn">UPLOAD AND SLICE</button>  
 </form>  
 <div class="platform-settings">  
 <div class="platform-settings-title">PLATFORM SETTINGS</div>  
 <form method="post" class="platform-form">  
 <div class="platform-form-group">  
 <label class="platform-form-label">Width (mm)</label>  
 <input type="number" name="platform\_width" min="100" max="500" step="1"  
 value="<?= $currentPlatformWidth ?>" required>  
 </div>  
 <div class="platform-form-group">  
 <label class="platform-form-label">Depth (mm)</label>  
 <input type="number" name="platform\_depth" min="100" max="500" step="1"  
 value="<?= $currentPlatformDepth ?>" required>  
 </div>  
 <div class="platform-form-group">  
 <label class="platform-form-label">Height (mm)</label>  
 <input type="number" name="platform\_height" min="100" max="500" step="1"  
 value="<?= $currentPlatformHeight ?>" required>  
 </div>  
 <button type="submit" name="save\_platform\_size" class="btn">Apply Settings</button>  
 </form>  
 </div>  
 </div>  
 <div class="card">  
 <h3 class="card-title">PRINT HISTORY</h3>  
 <?php if (empty($prints)): ?>  
 <div class="empty-state">No print history found</div>  
 <?php else: ?>  
 <div class="table-responsive">  
 <table class="print-table" style="width: 100%; border-collapse: collapse;">  
 <thead>  
 <tr style="background: rgba(10, 189, 198, 0.2);">  
 <th style="padding: 12px; text-align: left; border-bottom: 1px solid var(--primary-blue);">User</th>  
 <th style="padding: 12px; text-align: left; border-bottom: 1px solid var(--primary-blue);">Model</th>  
 <th style="padding: 12px; text-align: left; border-bottom: 1px solid var(--primary-blue);">Time</th>  
 <th style="padding: 12px; text-align: left; border-bottom: 1px solid var(--primary-blue);">Details</th>  
 </tr>  
 </thead>  
 <tbody>  
 <?php foreach ($prints as $print): ?>  
 <tr style="border-bottom: 1px solid rgba(100, 181, 246, 0.3);">  
 <td style="padding: 12px; color: var(--accent-blue);"><?= htmlspecialchars($print['username']) ?></td>  
 <td style="padding: 12px;"><?= htmlspecialchars($print['model\_name']) ?></td>  
 <td style="padding: 12px; color: var(--accent-blue);"><?= date('M d, Y H:i', strtotime($print['created\_at'])) ?></td>  
 <td style="padding: 12px;">  
 <span style="display: block; font-size: 0.8rem;">Material: <?= htmlspecialchars($print['material']) ?></span>  
 <span style="display: block; font-size: 0.8rem;">Time: <?= htmlspecialchars($print['print\_time']) ?></span>  
 <span style="display: block; font-size: 0.8rem;">Filament: <?= htmlspecialchars($print['filament\_used']) ?></span>  
 </td>  
 </tr>  
 <?php endforeach; ?>  
 </tbody>  
 </table>  
 </div>  
 <?php endif; ?>  
 </div>  
<script>  
 // Matrix effect  
 *document*.addEventListener('DOMContentLoaded', function() {  
 const canvas = *document*.createElement('canvas');  
 const container = *document*.getElementById('matrix-effect');  
 container.appendChild(canvas);  
 const ctx = canvas.getContext('2d');  
 canvas.width = container.offsetWidth;  
 canvas.height = container.offsetHeight;  
 const chars = "01アイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワヲン";  
 const fontSize = 14;  
 const columns = canvas.width / fontSize;  
 const drops = [];  
 for (let i = 0; i < columns; i++) {  
 drops[i] = *Math*.random() \* canvas.height;  
 }  
 function draw() {  
 ctx.fillStyle = 'rgba(0, 0, 0, 0.05)';  
 ctx.fillRect(0, 0, canvas.width, canvas.height);  
 ctx.fillStyle = '#0abdc6';  
 ctx.font = fontSize + 'px monospace';  
 for (let i = 0; i < drops.length; i++) {  
 const text = chars[*Math*.floor(*Math*.random() \* chars.length)];  
 ctx.fillText(text, i \* fontSize, drops[i] \* fontSize);  
 if (drops[i] \* fontSize > canvas.height && *Math*.random() > 0.975) {  
 drops[i] = 0;  
 }  
 drops[i]++;  
 }  
 }  
 setInterval(draw, 33);  
 // File input display  
 *document*.getElementById('model-file').addEventListener('change', function(e) {  
 if (this.files.length > 0) {  
 const fileName = this.files[0].name;  
 const uploadText = *document*.querySelector('.upload-text');  
 uploadText.textContent = fileName;  
 }  
 });  
 });  
</script>  
</body>  
</html>

download.php

<?php  
require\_once *\_\_DIR\_\_*.'/../includes/Config.php';  
require\_once *\_\_DIR\_\_*.'/../includes/Auth.php';  
$auth = new Auth();  
if (!$auth->isLoggedIn()) {  
 header('HTTP/1.1 403 Forbidden');  
 die('Not authorized');  
}  
$file = $\_GET['file'] ?? '';  
$filepath = *GCODE\_UPLOAD\_DIR* . $file;  
if (!file\_exists($filepath)) {  
 header('HTTP/1.1 404 Not Found');  
 die('File not found');  
}  
header('Content-Type: application/octet-stream');  
header('Content-Disposition: attachment; filename="' . basename($filepath) . '"');  
header('Content-Length: ' . filesize($filepath));  
readfile($filepath);  
exit;

index.php

<?php  
error\_reporting(*E\_ALL* & ~*E\_NOTICE*);  
require\_once *\_\_DIR\_\_*.'/includes/Auth.php';  
require\_once *\_\_DIR\_\_*.'/includes/Config.php';  
session\_start();  
$auth = new Auth();  
$error = '';  
if ($auth->isLoggedIn()) {  
 header("Location: dashboard.php");  
 exit;  
}  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 if (isset($\_POST['login'])) {  
 $email = $\_POST['email'] ?? '';  
 $password = $\_POST['password'] ?? '';  
 if ($auth->login($email, $password)) {  
 header("Location: dashboard.php");  
 exit;  
 } else {  
 $error = "Invalid email or password";  
 }  
 }  
 elseif (isset($\_POST['register'])) {  
 $username = $\_POST['username'] ?? '';  
 $email = $\_POST['email'] ?? '';  
 $password = $\_POST['password'] ?? '';  
 $confirm = $\_POST['confirm\_password'] ?? '';  
 if ($password !== $confirm) {  
 $error = "Passwords don't match";  
 } else {  
 if ($auth->register($username, $email, $password)) {  
 $error = "Registration successful! Please login.";  
 } else {  
 $error = "Registration failed. Email may already exist.";  
 }  
 }  
 }  
}  
?>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Lite Slicer - 3D Printer Web Slicer</title>  
 <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap" rel="stylesheet">  
 <style>  
 :root {  
 --primary-blue: #0abdc6;  
 --dark-blue: #1a237e;  
 --light-blue: #e3f2fd;  
 --accent-blue: #64b5f6;  
 --text-dark: #000000;  
 --text-light: #ffffff;  
 }  
 body {  
 margin: 0;  
 padding: 0;  
 font-family: 'Roboto', sans-serif;  
 background-color: #0c0c1a;  
 color: var(--text-light);  
 height: 100vh;  
 overflow: hidden;  
 position: relative;  
 }  
 #matrix-effect {  
 position: fixed;  
 top: 0;  
 left: 0;  
 z-index: -1;  
 opacity: 0.1;  
 width: 100%;  
 height: 100%;  
 }  
 .container {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 justify-content: center;  
 height: 100vh;  
 padding: 20px;  
 box-sizing: border-box;  
 }  
 .logo-container {  
 margin-bottom: 40px;  
 text-align: center;  
 }  
 .logo-img {  
 height: 80px;  
 margin-bottom: 15px;  
 }  
 .logo-text {  
 font-size: 2.5rem;  
 font-weight: 700;  
 background: linear-gradient(90deg, #64b5f6, #0abdc6);  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 margin-bottom: 5px;  
 }  
 .logo-subtext {  
 font-size: 1rem;  
 color: var(--accent-blue);  
 font-weight: 300;  
 }  
 .auth-box {  
 background: rgba(26, 35, 126, 0.7);  
 border: 1px solid var(--primary-blue);  
 border-radius: 10px;  
 padding: 30px;  
 width: 100%;  
 max-width: 400px;  
 box-shadow: 0 0 20px rgba(10, 189, 198, 0.3);  
 backdrop-filter: blur(5px);  
 }  
 .auth-tabs {  
 display: flex;  
 margin-bottom: 20px;  
 border-bottom: 1px solid var(--primary-blue);  
 }  
 .auth-tab {  
 padding: 10px 20px;  
 cursor: pointer;  
 color: var(--text-light);  
 font-weight: 500;  
 opacity: 0.7;  
 transition: all 0.3s;  
 }  
 .auth-tab.active {  
 opacity: 1;  
 border-bottom: 2px solid var(--primary-blue);  
 color: var(--primary-blue);  
 }  
 .auth-form {  
 display: none;  
 }  
 .auth-form.active {  
 display: block;  
 }  
 .form-group {  
 margin-bottom: 20px;  
 }  
 .form-group label {  
 display: block;  
 margin-bottom: 8px;  
 color: var(--accent-blue);  
 font-size: 0.9rem;  
 }  
 .form-control {  
 width: 100%;  
 padding: 12px;  
 border: 1px solid rgba(100, 181, 246, 0.3);  
 border-radius: 5px;  
 background: rgba(10, 189, 198, 0.1);  
 color: var(--text-light);  
 font-size: 1rem;  
 transition: all 0.3s;  
 }  
 .form-control:focus {  
 outline: none;  
 border-color: var(--primary-blue);  
 box-shadow: 0 0 10px rgba(10, 189, 198, 0.3);  
 }  
 .btn {  
 width: 100%;  
 padding: 12px;  
 border: none;  
 border-radius: 5px;  
 background: linear-gradient(90deg, var(--primary-blue), #64b5f6);  
 color: var(--text-dark);  
 font-weight: 700;  
 font-size: 1rem;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
 .btn:hover {  
 box-shadow: 0 0 15px rgba(10, 189, 198, 0.5);  
 }  
 .error-message {  
 color: #ff5252;  
 margin-bottom: 20px;  
 text-align: center;  
 font-size: 0.9rem;  
 }  
 .success-message {  
 color: #69f0ae;  
 margin-bottom: 20px;  
 text-align: center;  
 font-size: 0.9rem;  
 }  
 </style>  
</head>  
<body>  
<div id="matrix-effect"></div>  
<div class="container">  
 <div class="logo-container">  
 <img src="photo/photo2.png" alt="Lite Slicer Logo" class="logo-img">  
 <div class="logo-text">Lite Slicer</div>  
 <div class="logo-subtext">Professional 3D Printing Web Slicer</div>  
 </div>  
 <div class="auth-box">  
 <div class="auth-tabs">  
 <div class="auth-tab active" onclick="switchTab('login')">Login</div>  
 <div class="auth-tab" onclick="switchTab('register')">Register</div>  
 </div>  
 <?php if ($error): ?>  
 <div class="<?= strpos($error, 'success') !== false ? 'success-message' : 'error-message' ?>">  
 <?= htmlspecialchars($error) ?>  
 </div>  
 <?php endif; ?>  
 <form method="POST" action="" class="auth-form active" id="login-form">  
 <div class="form-group">  
 <label for="loginEmail">Email</label>  
 <input type="email" id="loginEmail" name="email" class="form-control" required>  
 </div>  
 <div class="form-group">  
 <label for="loginPassword">Password</label>  
 <input type="password" id="loginPassword" name="password" class="form-control" required>  
 </div>  
 <button type="submit" name="login" class="btn">Login</button>  
 </form>  
  
 <form method="POST" action="" class="auth-form" id="register-form">  
 <div class="form-group">  
 <label for="registerUsername">Username</label>  
 <input type="text" id="registerUsername" name="username" class="form-control" required>  
 </div>  
 <div class="form-group">  
 <label for="registerEmail">Email</label>  
 <input type="email" id="registerEmail" name="email" class="form-control" required>  
 </div>  
 <div class="form-group">  
 <label for="registerPassword">Password</label>  
 <input type="password" id="registerPassword" name="password" class="form-control" required>  
 </div>  
 <div class="form-group">  
 <label for="registerConfirm">Confirm Password</label>  
 <input type="password" id="registerConfirm" name="confirm\_password" class="form-control" required>  
 </div>  
 <button type="submit" name="register" class="btn">Register</button>  
 </form>  
 </div>  
</div>  
<script>  
 // Matrix effect  
 *document*.addEventListener('DOMContentLoaded', function() {  
 const canvas = *document*.createElement('canvas');  
 const container = *document*.getElementById('matrix-effect');  
 container.appendChild(canvas);  
 const ctx = canvas.getContext('2d');  
 canvas.width = container.offsetWidth;  
 canvas.height = container.offsetHeight;  
 const chars = "01アイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワヲン";  
 const fontSize = 14;  
 const columns = canvas.width / fontSize;  
 const drops = [];  
 for (let i = 0; i < columns; i++) {  
 drops[i] = *Math*.random() \* canvas.height;  
 }  
 function draw() {  
 ctx.fillStyle = 'rgba(0, 0, 0, 0.05)';  
 ctx.fillRect(0, 0, canvas.width, canvas.height);  
 ctx.fillStyle = '#0abdc6';  
 ctx.font = fontSize + 'px monospace';  
 for (let i = 0; i < drops.length; i++) {  
 const text = chars[*Math*.floor(*Math*.random() \* chars.length)];  
 ctx.fillText(text, i \* fontSize, drops[i] \* fontSize);  
 if (drops[i] \* fontSize > canvas.height && *Math*.random() > 0.975) {  
 drops[i] = 0;  
 }  
 drops[i]++;  
 }  
 }  
 setInterval(draw, 33);  
 });  
 function switchTab(tab) {  
 *document*.querySelectorAll('.auth-tab').forEach(t => t.classList.remove('active'));  
 *document*.querySelectorAll('.auth-form').forEach(f => f.classList.remove('active'));  
 *document*.querySelector(`.auth-tab[onclick="switchTab('${tab}')"]`).classList.add('active');  
 *document*.getElementById(`${tab}-form`).classList.add('active');  
 }  
</script>  
</body>  
</html>

login.php

<?php  
require\_once *\_\_DIR\_\_*.'/includes/Auth.php';  
require\_once *\_\_DIR\_\_*.'/includes/Database.php';  
$auth = new Auth();  
if ($auth->isLoggedIn()) {  
 header("Location: dashboard.php");  
 exit;  
}  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 $email = $\_POST['email'] ?? '';  
 $password = $\_POST['password'] ?? '';  
 if ($auth->login($email, $password)) {  
 header("Location: dashboard.php");  
 exit;  
 } else {  
 $error = "Invalid email or password";  
 }  
}  
include *\_\_DIR\_\_*.'/templates/header.php';  
include *\_\_DIR\_\_*.'/templates/login.php';  
include *\_\_DIR\_\_*.'/templates/footer.php';

logout.php

<?php  
require\_once *\_\_DIR\_\_* . '/includes/Auth.php';  
$auth = new Auth();  
$auth->logout();  
header("Location: login.php");  
exit;  
?>

package.json

{  
 "dependencies": {  
 "@popperjs/core": "^2.11.8",  
 "@types/three": "^0.176.0",  
 "bootstrap": "^5.3.5",  
 "bootstrap-icons": "^1.11.3",  
 "three": "^0.175.0"  
 },  
 "name": "3d-printer-slicer",  
 "version": "1.0.0",  
 "main": "index.js",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1"  
 },  
 "keywords": [],  
 "author": "",  
 "license": "ISC",  
 "description": ""  
}

package-lock.json

{  
 "name": "3d-printer-slicer",  
 "version": "1.0.0",  
 "lockfileVersion": 3,  
 "requires": true,  
 "packages": {  
 "": {  
 "name": "3d-printer-slicer",  
 "version": "1.0.0",  
 "license": "ISC",  
 "dependencies": {  
 "@popperjs/core": "^2.11.8",  
 "@types/three": "^0.176.0",  
 "bootstrap": "^5.3.5",  
 "bootstrap-icons": "^1.11.3",  
 "three": "^0.175.0"  
 }  
 },  
 "node\_modules/@dimforge/rapier3d-compat": {  
 "version": "0.12.0",  
 "resolved": "https://registry.npmjs.org/@dimforge/rapier3d-compat/-/rapier3d-compat-0.12.0.tgz",  
 "integrity": "sha512-uekIGetywIgopfD97oDL5PfeezkFpNhwlzlaEYNOA0N6ghdsOvh/HYjSMek5Q2O1PYvRSDFcqFVJl4r4ZBwOow==",  
 "license": "Apache-2.0"  
 },  
 "node\_modules/@popperjs/core": {  
 "version": "2.11.8",  
 "resolved": "https://registry.npmjs.org/@popperjs/core/-/core-2.11.8.tgz",  
 "integrity": "sha512-P1st0aksCrn9sGZhp8GMYwBnQsbvAWsZAX44oXNNvLHGqAOcoVxmjZiohstwQ7SqKnbR47akdNi+uleWD8+g6A==",  
 "license": "MIT",  
 "funding": {  
 "type": "opencollective",  
 "url": "https://opencollective.com/popperjs"  
 }  
 },  
 "node\_modules/@tweenjs/tween.js": {  
 "version": "23.1.3",  
 "resolved": "https://registry.npmjs.org/@tweenjs/tween.js/-/tween.js-23.1.3.tgz",  
 "integrity": "sha512-vJmvvwFxYuGnF2axRtPYocag6Clbb5YS7kLL+SO/TeVFzHqDIWrNKYtcsPMibjDx9O+bu+psAy9NKfWklassUA==",  
 "license": "MIT"  
 },  
 "node\_modules/@types/stats.js": {  
 "version": "0.17.3",  
 "resolved": "https://registry.npmjs.org/@types/stats.js/-/stats.js-0.17.3.tgz",  
 "integrity": "sha512-pXNfAD3KHOdif9EQXZ9deK82HVNaXP5ZIF5RP2QG6OQFNTaY2YIetfrE9t528vEreGQvEPRDDc8muaoYeK0SxQ==",  
 "license": "MIT"  
 },  
 "node\_modules/@types/three": {  
 "version": "0.176.0",  
 "resolved": "https://registry.npmjs.org/@types/three/-/three-0.176.0.tgz",  
 "integrity": "sha512-FwfPXxCqOtP7EdYMagCFePNKoG1AGBDUEVKtluv2BTVRpSt7b+X27xNsirPCTCqY1pGYsPUzaM3jgWP7dXSxlw==",  
 "license": "MIT",  
 "dependencies": {  
 "@dimforge/rapier3d-compat": "^0.12.0",  
 "@tweenjs/tween.js": "~23.1.3",  
 "@types/stats.js": "\*",  
 "@types/webxr": "\*",  
 "@webgpu/types": "\*",  
 "fflate": "~0.8.2",  
 "meshoptimizer": "~0.18.1"  
 }  
 },  
 "node\_modules/@types/webxr": {  
 "version": "0.5.22",  
 "resolved": "https://registry.npmjs.org/@types/webxr/-/webxr-0.5.22.tgz",  
 "integrity": "sha512-Vr6Stjv5jPRqH690f5I5GLjVk8GSsoQSYJ2FVd/3jJF7KaqfwPi3ehfBS96mlQ2kPCwZaX6U0rG2+NGHBKkA/A==",  
 "license": "MIT"  
 },  
 "node\_modules/@webgpu/types": {  
 "version": "0.1.60",  
 "resolved": "https://registry.npmjs.org/@webgpu/types/-/types-0.1.60.tgz",  
 "integrity": "sha512-8B/tdfRFKdrnejqmvq95ogp8tf52oZ51p3f4QD5m5Paey/qlX4Rhhy5Y8tgFMi7Ms70HzcMMw3EQjH/jdhTwlA==",  
 "license": "BSD-3-Clause"  
 },  
 "node\_modules/bootstrap": {  
 "version": "5.3.5",  
 "resolved": "https://registry.npmjs.org/bootstrap/-/bootstrap-5.3.5.tgz",  
 "integrity": "sha512-ct1CHKtiobRimyGzmsSldEtM03E8fcEX4Tb3dGXz1V8faRwM50+vfHwTzOxB3IlKO7m+9vTH3s/3C6T2EAPeTA==",  
 "funding": [  
 {  
 "type": "github",  
 "url": "https://github.com/sponsors/twbs"  
 },  
 {  
 "type": "opencollective",  
 "url": "https://opencollective.com/bootstrap"  
 }  
 ],  
 "license": "MIT",  
 "peerDependencies": {  
 "@popperjs/core": "^2.11.8"  
 }  
 },  
 "node\_modules/bootstrap-icons": {  
 "version": "1.11.3",  
 "resolved": "https://registry.npmjs.org/bootstrap-icons/-/bootstrap-icons-1.11.3.tgz",  
 "integrity": "sha512-+3lpHrCw/it2/7lBL15VR0HEumaBss0+f/Lb6ZvHISn1mlK83jjFpooTLsMWbIjJMDjDjOExMsTxnXSIT4k4ww==",  
 "funding": [  
 {  
 "type": "github",  
 "url": "https://github.com/sponsors/twbs"  
 },  
 {  
 "type": "opencollective",  
 "url": "https://opencollective.com/bootstrap"  
 }  
 ],  
 "license": "MIT"  
 },  
 "node\_modules/fflate": {  
 "version": "0.8.2",  
 "resolved": "https://registry.npmjs.org/fflate/-/fflate-0.8.2.tgz",  
 "integrity": "sha512-cPJU47OaAoCbg0pBvzsgpTPhmhqI5eJjh/JIu8tPj5q+T7iLvW/JAYUqmE7KOB4R1ZyEhzBaIQpQpardBF5z8A==",  
 "license": "MIT"  
 },  
 "node\_modules/meshoptimizer": {  
 "version": "0.18.1",  
 "resolved": "https://registry.npmjs.org/meshoptimizer/-/meshoptimizer-0.18.1.tgz",  
 "integrity": "sha512-ZhoIoL7TNV4s5B6+rx5mC//fw8/POGyNxS/DZyCJeiZ12ScLfVwRE/GfsxwiTkMYYD5DmK2/JXnEVXqL4rF+Sw==",  
 "license": "MIT"  
 },  
 "node\_modules/three": {  
 "version": "0.175.0",  
 "resolved": "https://registry.npmjs.org/three/-/three-0.175.0.tgz",  
 "integrity": "sha512-nNE3pnTHxXN/Phw768u0Grr7W4+rumGg/H6PgeseNJojkJtmeHJfZWi41Gp2mpXl1pg1pf1zjwR4McM1jTqkpg==",  
 "license": "MIT"  
 }  
 }  
}

register.php

<?php  
require\_once *\_\_DIR\_\_*.'/includes/Auth.php';  
$auth = new Auth();  
if ($auth->isLoggedIn()) {  
 header("Location: dashboard.php");  
 exit;  
}  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 $username = $\_POST['username'] ?? '';  
 $email = $\_POST['email'] ?? '';  
 $password = $\_POST['password'] ?? '';  
 $confirm\_password = $\_POST['confirm\_password'] ?? '';  
  
 if ($password !== $confirm\_password) {  
 $error = "Passwords don't match";  
 } elseif ($auth->register($username, $email, $password)) {  
 header("Location: login.php?registered=1");  
 exit;  
 } else {  
 $error = "Registration failed. Email may already exist.";  
 }  
}  
include *\_\_DIR\_\_*.'/templates/header.php';  
include *\_\_DIR\_\_*.'/templates/register.php';  
include *\_\_DIR\_\_*.'/templates/footer.php';

slicer.php

<?php  
require\_once *\_\_DIR\_\_*.'/includes/Config.php';  
require\_once *\_\_DIR\_\_*.'/includes/Auth.php';  
require\_once *\_\_DIR\_\_*.'/includes/FileUploader.php';  
$auth = new Auth();  
if (!$auth->isLoggedIn()) {  
 header("Location: login.php");  
 exit;  
}  
$modelFile = $\_GET['model'] ?? '';  
$modelPath = *MODEL\_UPLOAD\_DIR* . $modelFile;  
if (!file\_exists($modelPath)) {  
 header("Location: dashboard.php");  
 exit;  
}  
$modelInfo = FileUploader::*getModelInfo*($modelFile);  
?>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Lite Slicer - 3D Slicer</title>  
 <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap" rel="stylesheet">  
 <style>  
 :root {  
 --primary-blue: #0abdc6;  
 --dark-blue: #1a237e;  
 --light-blue: #e3f2fd;  
 --accent-blue: #64b5f6;  
 --taext-dark: #212121;  
 --text-light: #f5f5f5;  
 --card-bg: rgba(26, 35, 126, 0.7);  
 }  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
 body {  
 font-family: 'Roboto', sans-serif;  
 background-color: #0c0c1a;  
 color: var(--text-light);  
 height: 100vh;  
 overflow: hidden;  
 }  
 #matrix-effect {  
 position: fixed;  
 top: 0;  
 left: 0;  
 z-index: -1;  
 opacity: 0.1;  
 width: 100%;  
 height: 100%;  
 }  
 .app-header {  
 display: flex;  
 align-items: center;  
 padding: 15px 20px;  
 background: var(--card-bg);  
 border-bottom: 1px solid var(--primary-blue);  
 backdrop-filter: blur(5px);  
 }  
 .logo {  
 display: flex;  
 align-items: center;  
 }  
 .logo img {  
 height: 30px;  
 margin-right: 10px;  
 }  
 .logo-text {  
 font-size: 1.2rem;  
 font-weight: 700;  
 background: linear-gradient(90deg, #64b5f6, #0abdc6);  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 }  
 .user-menu {  
 margin-left: auto;  
 display: flex;  
 align-items: center;  
 }  
 .user-avatar {  
 width: 32px;  
 height: 32px;  
 border-radius: 50%;  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-weight: 700;  
 margin-right: 10px;  
 }  
 .main-container {  
 display: grid;  
 grid-template-columns: 300px 1fr;  
 height: calc(100vh - 62px);  
 }  
 .sidebar {  
 background: var(--card-bg);  
 border-right: 1px solid var(--primary-blue);  
 padding: 20px;  
 overflow-y: auto;  
 backdrop-filter: blur(5px);  
 }  
 .section {  
 margin-bottom: 30px;  
 }  
 .section-title {  
 color: var(--primary-blue);  
 font-size: 1.1rem;  
 margin-bottom: 15px;  
 padding-bottom: 8px;  
 border-bottom: 1px solid rgba(100, 181, 246, 0.3);  
 }  
 .model-info-grid {  
 display: grid;  
 grid-template-columns: 1fr 1fr;  
 gap: 12px;  
 }  
 .info-item {  
 margin-bottom: 12px;  
 }  
 .info-label {  
 color: var(--accent-blue);  
 font-size: 0.8rem;  
 margin-bottom: 4px;  
 }  
 .info-value {  
 font-size: 0.9rem;  
 word-break: break-all;  
 }  
 .form-group {  
 margin-bottom: 15px;  
 }  
 .form-label {  
 display: block;  
 color: var(--accent-blue);  
 font-size: 0.9rem;  
 margin-bottom: 6px;  
 }  
 .form-control {  
 width: 100%;  
 padding: 8px 12px;  
 background: rgba(10, 189, 198, 0.1);  
 border: 1px solid rgba(100, 181, 246, 0.3);  
 border-radius: 5px;  
 color: var(--text-light);  
 font-family: inherit;  
 }  
 .form-control:focus {  
 outline: none;  
 border-color: var(--primary-blue);  
 box-shadow: 0 0 0 2px rgba(10, 189, 198, 0.2);  
 }  
 .slider-container {  
 margin-bottom: 20px;  
 }  
 .slider-label {  
 display: flex;  
 justify-content: space-between;  
 margin-bottom: 5px;  
 color: var(--accent-blue);  
 font-size: 0.9rem;  
 }  
 .slider {  
 width: 100%;  
 height: 6px;  
 -webkit-appearance: none;  
 background: linear-gradient(90deg, var(--primary-blue), var(--accent-blue));  
 border-radius: 3px;  
 outline: none;  
 }  
 .slider::-webkit-slider-thumb {  
 -webkit-appearance: none;  
 width: 16px;  
 height: 16px;  
 border-radius: 50%;  
 background: var(--primary-blue);  
 cursor: pointer;  
 }  
 .checkbox-group {  
 display: flex;  
 align-items: center;  
 margin-bottom: 12px;  
 }  
 .checkbox-group input {  
 margin-right: 8px;  
 accent-color: var(--primary-blue);  
 }  
 .btn {  
 width: 100%;  
 padding: 10px;  
 border: none;  
 border-radius: 5px;  
 background: linear-gradient(90deg, var(--primary-blue), var(--accent-blue));  
 color: var(--text-dark);  
 font-weight: 700;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
 .btn:hover {  
 box-shadow: 0 0 15px rgba(10, 189, 198, 0.5);  
 }  
 .gcode-terminal {  
 background: rgba(0, 0, 0, 0.3);  
 border: 1px solid var(--primary-blue);  
 border-radius: 5px;  
 padding: 12px;  
 height: 200px;  
 overflow-y: auto;  
 font-family: 'Courier New', monospace;  
 font-size: 0.85rem;  
 color: var(--accent-blue);  
 line-height: 1.4;  
 }  
 .terminal-line {  
 margin-bottom: 5px;  
 }  
 .terminal-line::before {  
 content: ">";  
 margin-right: 5px;  
 color: var(--primary-blue);  
 }  
 .download-btn {  
 width: 100%;  
 padding: 10px;  
 margin-top: 15px;  
 border: none;  
 border-radius: 5px;  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 font-weight: 700;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
 .download-btn:hover {  
 box-shadow: 0 0 15px rgba(10, 189, 198, 0.5);  
 }  
 .download-btn.disabled {  
 opacity: 0.5;  
 cursor: not-allowed;  
 background: #555;  
 }  
 .loading {  
 display: none;  
 text-align: center;  
 color: var(--primary-blue);  
 margin: 15px 0;  
 }  
 .spinner {  
 display: inline-block;  
 width: 18px;  
 height: 18px;  
 border: 3px solid rgba(100, 181, 246, 0.3);  
 border-radius: 50%;  
 border-top-color: var(--primary-blue);  
 animation: spin 1s linear infinite;  
 margin-right: 8px;  
 vertical-align: middle;  
 }  
 @keyframes spin {  
 to { transform: rotate(360deg); }  
 }  
 .viewer-container {  
 display: flex;  
 flex-direction: column;  
 height: 100%;  
 }  
 .viewer-toolbar {  
 padding: 10px 15px;  
 background: var(--card-bg);  
 border-bottom: 1px solid var(--primary-blue);  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
 .viewer-title {  
 color: var(--primary-blue);  
 font-weight: 500;  
 }  
 .toolbar-actions {  
 display: flex;  
 gap: 8px;  
 }  
 .toolbar-btn {  
 background: rgba(10, 189, 198, 0.2);  
 border: 1px solid var(--primary-blue);  
 color: var(--primary-blue);  
 padding: 6px 10px;  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 0.85rem;  
 transition: all 0.3s;  
 }  
 .toolbar-btn:hover {  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 }  
 #viewer-wrapper {  
 flex: 1;  
 position: relative;  
 }  
 #viewer {  
 width: 100%;  
 height: 100%;  
 }  
 .control-panel {  
 position: absolute;  
 bottom: 15px;  
 right: 15px;  
 z-index: 100;  
 }  
 .control-btn {  
 width: 36px;  
 height: 36px;  
 border-radius: 50%;  
 background: rgba(0, 0, 0, 0.7);  
 border: 1px solid var(--primary-blue);  
 color: var(--primary-blue);  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 cursor: pointer;  
 font-size: 16px;  
 transition: all 0.3s;  
 margin-bottom: 5px;  
 }  
 .control-btn:hover {  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 transform: scale(1.1);  
 }  
 .control-group {  
 display: flex;  
 gap: 5px;  
 margin-bottom: 5px;  
 }  
 .rotate-btn {  
 background: rgba(10, 189, 198, 0.2);  
 border: 1px solid var(--primary-blue);  
 color: var(--primary-blue);  
 padding: 6px 10px;  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 0.85rem;  
 transition: all 0.3s;  
 }  
 .rotate-btn:hover {  
 background: var(--primary-blue);  
 color: var(--text-dark);  
 }  
 </style>  
</head>  
<body>  
<div id="matrix-effect"></div>  
<header class="app-header">  
 <div class="logo">  
 <img src="photo/photo1.jpg" alt="Lite Slicer Logo">  
 <div class="logo-text">Lite Slicer</div>  
 </div>  
 <div class="user-menu">  
 <div class="user-avatar"><?= strtoupper(substr($auth->getCurrentUser()['username'], 0, 1)) ?></div>  
 <a href="dashboard.php" class="toolbar-btn">Dashboard</a>  
 </div>  
</header>  
<div class="main-container">  
 <div class="sidebar">  
 <div class="section">  
 <h3 class="section-title">MODEL INFORMATION</h3>  
 <div class="model-info-grid">  
 <div class="info-item">  
 <div class="info-label">File Name</div>  
 <div class="info-value"><?= htmlspecialchars($modelInfo['filename']) ?></div>  
 </div>  
 <div class="info-item">  
 <div class="info-label">File Size</div>  
 <div class="info-value"><?= htmlspecialchars($modelInfo['size']) ?></div>  
 </div>  
 <div class="info-item">  
 <div class="info-label">Dimensions</div>  
 <div class="info-value" id="model-dimensions"><?= htmlspecialchars($modelInfo['dimensions']) ?></div>  
 </div>  
 <div class="info-item">  
 <div class="info-label">Volume</div>  
 <div class="info-value" id="model-volume"><?= htmlspecialchars($modelInfo['volume']) ?></div>  
 </div>  
 </div>  
 </div>  
 <div class="section">  
 <h3 class="section-title">SLICING SETTINGS</h3>  
 <form id="slice-form">  
 <input type="hidden" name="model" value="<?= htmlspecialchars($modelFile) ?>">  
  
 <div class="form-group">  
 <label class="form-label">Material</label>  
 <select name="material" class="form-control">  
 <option value="PLA" selected>PLA</option>  
 <option value="ABS">ABS</option>  
 <option value="PETG">PETG</option>  
 <option value="TPU">TPU</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label class="form-label">Nozzle Size (mm)</label>  
 <select name="nozzle\_size" class="form-control">  
 <option value="0.2">0.2</option>  
 <option value="0.4" selected>0.4</option>  
 <option value="0.6">0.6</option>  
 <option value="0.8">0.8</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label class="form-label">Layer Height (mm)</label>  
 <input type="number" name="layer\_height" min="0.05" max="0.3" step="0.05" value="0.2" class="form-control">  
 </div>  
 <div class="slider-container">  
 <div class="slider-label">  
 <span>Infill Density</span>  
 <span id="infill-value">20%</span>  
 </div>  
 <input type="range" id="infill-density" name="infill\_density" min="0" max="100" value="20" class="slider">  
 </div>  
 <div class="checkbox-group">  
 <input type="checkbox" id="generate-support" name="generate\_support">  
 <label for="generate-support">Generate Support</label>  
 </div>  
 <div class="checkbox-group">  
 <input type="checkbox" id="add-brim" name="add\_brim">  
 <label for="add-brim">Add Brim</label>  
 </div>  
 <button type="submit" id="slice-btn" class="btn">SLICE MODEL</button>  
 </form>  
 </div>  
 <div class="scale-controls">  
 </div>  
 <div class="section">  
 <h3 class="section-title">G-CODE OUTPUT</h3>  
 <div class="gcode-terminal" id="gcode-terminal">  
 <div class="terminal-line">Ready to slice model...</div>  
 </div>  
 <button id="download-btn" class="download-btn disabled">DOWNLOAD G-CODE</button>  
 <div class="loading" id="loading">  
 <span class="spinner"></span>  
 <span>Processing...</span>  
 </div>  
 </div>  
 </div>  
 <div class="viewer-container">  
 <div class="viewer-toolbar">  
 <div class="viewer-title">3D MODEL VIEWER</div>  
 <div class="toolbar-actions">  
 <button class="toolbar-btn" id="reset-view">Reset View</button>  
 <button class="rotate-btn" id="rotate-left" title="Rotate Left">←</button>  
 <button class="rotate-btn" id="rotate-right" title="Rotate Right">→</button>  
 <button class="rotate-btn" id="rotate-up" title="Rotate Up">↑</button>  
 <button class="rotate-btn" id="rotate-down" title="Rotate Down">↓</button>  
 </div>  
 </div>  
 <div id="viewer-wrapper">  
 <div id="viewer"></div>  
 <div class="control-panel">  
 <div class="control-group">  
 <button class="control-btn" id="zoom-in" title="Zoom In">+</button>  
 <button class="control-btn" id="zoom-out" title="Zoom Out">-</button>  
 </div>  
 <div class="control-group">  
 <button class="control-btn" id="move-x-plus" title="Move X+">+X</button>  
 <button class="control-btn" id="move-x-minus" title="Move X-">-X</button>  
 </div>  
 <div class="control-group">  
 <button class="control-btn" id="move-y-plus" title="Move Y+">+Y</button>  
 <button class="control-btn" id="move-y-minus" title="Move Y-">-Y</button>  
 </div>  
 <div class="control-group">  
 <button class="control-btn" id="move-z-plus" title="Move Z+">+Z</button>  
 <button class="control-btn" id="move-z-minus" title="Move Z-">-Z</button>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
<script src="https://cdn.jsdelivr.net/npm/three@0.132.2/build/three.min.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/controls/OrbitControls.js"></script>  
<script src="https://cdn.jsdelivr.net/npm/three@0.132.2/examples/js/loaders/STLLoader.js"></script>  
<script>  
 // Matrix effect  
 *document*.addEventListener('DOMContentLoaded', function() {  
 const canvas = *document*.createElement('canvas');  
 const container = *document*.getElementById('matrix-effect');  
 container.appendChild(canvas);  
 const ctx = canvas.getContext('2d');  
 canvas.width = container.offsetWidth;  
 canvas.height = container.offsetHeight;  
 const chars = "01アイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワヲン";  
 const fontSize = 14;  
 const columns = canvas.width / fontSize;  
 const drops = [];  
 for (let i = 0; i < columns; i++) {  
 drops[i] = *Math*.random() \* canvas.height;  
 }  
 function draw() {  
 ctx.fillStyle = 'rgba(0, 0, 0, 0.05)';  
 ctx.fillRect(0, 0, canvas.width, canvas.height);  
 ctx.fillStyle = '#0abdc6';  
 ctx.font = fontSize + 'px monospace';  
 for (let i = 0; i < drops.length; i++) {  
 const text = chars[*Math*.floor(*Math*.random() \* chars.length)];  
 ctx.fillText(text, i \* fontSize, drops[i] \* fontSize);  
  
 if (drops[i] \* fontSize > canvas.height && *Math*.random() > 0.975) {  
 drops[i] = 0;  
 }  
 drops[i]++;  
 }  
 }  
 setInterval(draw, 33);  
 });  
 // 3D Viewer implementation  
 let *scene*, *camera*, *renderer*, *controls*, *model*;  
 let *modelCenter* = new THREE.Vector3();  
 const *ROTATION\_STEP* = 0.1; // Rotation step in radians  
 function initScene() {  
 const container = *document*.getElementById('viewer-wrapper');  
 // Scene setup  
 *scene* = new THREE.Scene();  
 *scene*.background = new THREE.Color(0x111126);  
 // Camera setup  
 *camera* = new THREE.PerspectiveCamera(  
 75,  
 container.clientWidth / container.clientHeight,  
 0.1,  
 1000  
 );  
 *camera*.position.z = 5;  
 // Renderer setup  
 *renderer* = new THREE.WebGLRenderer({ antialias: true });  
 *renderer*.setPixelRatio(*window*.devicePixelRatio);  
 *renderer*.setSize(container.clientWidth, container.clientHeight);  
 *renderer*.shadowMap.enabled = true;  
 *document*.getElementById('viewer').appendChild(*renderer*.domElement);  
 // Lighting  
 const ambientLight = new THREE.AmbientLight(0x404040);  
 *scene*.add(ambientLight);  
 const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);  
 directionalLight.position.set(1, 1, 1);  
 directionalLight.castShadow = true;  
 *scene*.add(directionalLight);  
 // Grid helper  
 const gridHelper = new THREE.GridHelper(200, 50, 0x003366, 0x002244);  
 gridHelper.position.y = -0.5;  
 *scene*.add(gridHelper);  
 // Load model  
 loadModel(`/uploads/models/<?= $modelFile ?>`);  
 // Controls  
 *controls* = new THREE.OrbitControls(*camera*, *renderer*.domElement);  
 *controls*.enableDamping = true;  
 *controls*.dampingFactor = 0.05;  
 *controls*.screenSpacePanning = false;  
 *controls*.maxPolarAngle = *Math*.PI;  
 *controls*.minPolarAngle = 0;  
 *controls*.enablePan = true;  
 // Event listeners  
 setupEventListeners();  
 // Handle window resize  
 *window*.addEventListener('resize', onWindowResize);  
 // Start animation loop  
 animate();  
 }  
 function loadModel(url) {  
 const loader = new THREE.STLLoader();  
 loader.load(url, function(geometry) {  
 if (*model*) *scene*.remove(*model*);  
 // Create material with neon blue color  
 const material = new THREE.MeshPhongMaterial({  
 color: 0x0abdc6,  
 specular: 0x111111,  
 shininess: 30,  
 side: THREE.DoubleSide,  
 flatShading: true  
 });  
 *model* = new THREE.Mesh(geometry, material);  
 *model*.castShadow = true;  
 // Center and scale model  
 geometry.computeBoundingBox();  
 const boundingBox = geometry.boundingBox;  
 boundingBox.getCenter(*modelCenter*);  
 *model*.position.sub(*modelCenter*);  
 const size = boundingBox.getSize(new THREE.Vector3());  
 const maxDim = *Math*.max(size.x, size.y, size.z);  
 const scale = 5 / maxDim;  
 *model*.scale.set(scale, scale, scale);  
 *scene*.add(*model*);  
 // Update model info  
 *document*.getElementById('model-dimensions').textContent =  
 `${(size.x \* scale).toFixed(1)} × ${(size.y \* scale).toFixed(1)} × ${(size.z \* scale).toFixed(1)} mm`;  
 const volume = (size.x \* size.y \* size.z \* scale \* scale \* scale / 1000).toFixed(1);  
 *document*.getElementById('model-volume').textContent = `${volume} cm³`;  
 // Adjust camera to fit model  
 fitCameraToModel(boundingBox, scale);  
 }, undefined, function(error) {  
 *console*.error('Error loading model:', error);  
 const terminal = *document*.getElementById('gcode-terminal');  
 terminal.innerHTML = '';  
 const errorLine = *document*.createElement('div');  
 errorLine.className = 'terminal-line';  
 errorLine.style.color = '#ff5252';  
 errorLine.textContent = `> ERROR: Failed to load model (${error})`;  
 terminal.appendChild(errorLine);  
 });  
 }  
 function fitCameraToModel(boundingBox, scale) {  
 const size = boundingBox.getSize(new THREE.Vector3());  
 const center = boundingBox.getCenter(new THREE.Vector3());  
 const maxDim = *Math*.max(size.x, size.y, size.z);  
 const fov = *camera*.fov \* (*Math*.PI / 180);  
 let cameraZ = *Math*.abs(maxDim \* scale / *Math*.sin(fov / 2)) \* 1.1;  
 cameraZ = *Math*.max(cameraZ, maxDim \* scale \* 0.5);  
 *camera*.position.z = cameraZ;  
 *camera*.lookAt(center);  
 *controls*.target.copy(center);  
 *controls*.update();  
 }  
 function onWindowResize() {  
 const container = *document*.getElementById('viewer-wrapper');  
 *camera*.aspect = container.clientWidth / container.clientHeight;  
 *camera*.updateProjectionMatrix();  
 *renderer*.setSize(container.clientWidth, container.clientHeight);  
 }  
 function animate() {  
 requestAnimationFrame(animate);  
 *controls*.update();  
 *renderer*.render(*scene*, *camera*);  
 }  
 function setupEventListeners() {  
 // Infill slider  
 *document*.getElementById('infill-density').addEventListener('input', function() {  
 *document*.getElementById('infill-value').textContent = `${this.value}%`;  
 });  
 // Zoom buttons  
 *document*.getElementById('zoom-in').addEventListener('click', () => {  
 *camera*.zoom \*= 1.2;  
 *camera*.updateProjectionMatrix();  
 });  
 *document*.getElementById('zoom-out').addEventListener('click', () => {  
 *camera*.zoom /= 1.2;  
 *camera*.updateProjectionMatrix();  
 });  
 // Move buttons (positive and negative directions)  
 ['x', 'y', 'z'].forEach(axis => {  
 // Positive direction  
 *document*.getElementById(`move-${axis}-plus`).addEventListener('click', () => {  
 if (*model*) {  
 const position = new THREE.Vector3();  
 position[axis] = 0.5;  
 *model*.position.add(position);  
 }  
 });  
 // Negative direction  
 *document*.getElementById(`move-${axis}-minus`).addEventListener('click', () => {  
 if (*model*) {  
 const position = new THREE.Vector3();  
 position[axis] = -0.5;  
 *model*.position.add(position);  
 }  
 });  
 });  
 // Rotation buttons  
 *document*.getElementById('rotate-left').addEventListener('click', () => {  
 if (*model*) {  
 *model*.rotation.y += *ROTATION\_STEP*;  
 }  
 });  
 *document*.getElementById('rotate-right').addEventListener('click', () => {  
 if (*model*) {  
 *model*.rotation.y -= *ROTATION\_STEP*;  
 }  
 });  
 *document*.getElementById('rotate-up').addEventListener('click', () => {  
 if (*model*) {  
 *model*.rotation.x += *ROTATION\_STEP*;  
 }  
 });  
 *document*.getElementById('rotate-down').addEventListener('click', () => {  
 if (*model*) {  
 *model*.rotation.x -= *ROTATION\_STEP*;  
 }  
 });  
 // Reset view  
 *document*.getElementById('reset-view').addEventListener('click', () => {  
 if (*model*) {  
 *controls*.reset();  
 *camera*.zoom = 1;  
 *camera*.updateProjectionMatrix();  
 *model*.rotation.set(0, 0, 0);  
 *camera*.lookAt(*modelCenter*);  
 *controls*.target.copy(*modelCenter*);  
 *controls*.update();  
 }  
 });  
 // Slice form submission  
 *document*.getElementById('slice-form').addEventListener('submit', function(e) {  
 e.preventDefault();  
 // Show loading state  
 *document*.getElementById('loading').style.display = 'block';  
 *document*.getElementById('slice-btn').disabled = true;  
 const terminal = *document*.getElementById('gcode-terminal');  
 terminal.innerHTML = '';  
 const statusLine = *document*.createElement('div');  
 statusLine.className = 'terminal-line';  
 statusLine.style.color = 'var(--primary-blue)';  
 statusLine.textContent = '> Starting slicing process...';  
 terminal.appendChild(statusLine);  
 // Collect form data  
 const formData = new FormData(this);  
 fetch('/slice.php', {  
 method: 'POST',  
 body: formData  
 })  
 .then(response => {  
 if (!response.ok) throw new Error('Network error');  
 return response.json();  
 })  
 .then(data => {  
 *document*.getElementById('loading').style.display = 'none';  
 *document*.getElementById('slice-btn').disabled = false;  
 if (data.success) {  
 // Clear terminal  
 terminal.innerHTML = '';  
 // Show G-code preview  
 const lines = data.gcode.split('\n');  
 const previewLines = lines.slice(0, 60);  
 previewLines.forEach(line => {  
 const lineElement = *document*.createElement('div');  
 lineElement.className = 'terminal-line';  
 lineElement.textContent = line;  
 terminal.appendChild(lineElement);  
 });  
 // Add info about full file  
 const infoLine = *document*.createElement('div');  
 infoLine.className = 'terminal-line';  
 infoLine.style.color = 'var(--primary-blue)';  
 infoLine.textContent = `> G-code generated (${lines.length} lines total)`;  
 terminal.appendChild(infoLine);  
 // Add print info  
 const printInfo = *document*.createElement('div');  
 printInfo.className = 'terminal-line';  
 printInfo.textContent = `> Estimated print time: ${data.print\_time}`;  
 terminal.appendChild(printInfo);  
 const filamentInfo = *document*.createElement('div');  
 filamentInfo.className = 'terminal-line';  
 filamentInfo.textContent = `> Filament used: ${data.filament\_used}`;  
 terminal.appendChild(filamentInfo);  
 // Enable download button  
 const downloadBtn = *document*.getElementById('download-btn');  
 downloadBtn.classList.remove('disabled');  
 downloadBtn.onclick = () => {  
 *window*.location.href = `/download.php?file=${encodeURIComponent(data.filename)}`;  
 };  
 } else {  
 showError(data.error || 'Unknown error occurred');  
 }  
 })  
 .catch(error => {  
 *document*.getElementById('loading').style.display = 'none';  
 *document*.getElementById('slice-btn').disabled = false;  
 showError(error.message);  
 });  
 function showError(message) {  
 const errorLine = *document*.createElement('div');  
 errorLine.className = 'terminal-line';  
 errorLine.style.color = '#ff5252';  
 errorLine.textContent = `> ERROR: ${message}`;  
 terminal.appendChild(errorLine);  
 }  
 });  
 }  
 // Initialize  
 *document*.addEventListener('DOMContentLoaded', initScene);  
</script>  
</body>  
</html>