

API_FUNCTIONS

[Publish.js](#)

```
// /api/publish.js
import { createClient } from '@supabase/supabase-js';
import crypto from 'crypto';

const supabase = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_ANON_KEY
);

// Enhanced rate limiting with exponential backoff
const rateLimitStore = new Map();
const RATE_LIMIT_CONFIG = {
  WINDOW_MS: 60 * 1000,
  MAX_REQUESTS: 10,
  BAN_WINDOW_MS: 15 * 60 * 1000,
  BAN_THRESHOLD: 20
};

// Allowed origins
const ALLOWED_ORIGINS = [
  'https://pack-cdn.vercel.app',
  'https://pack-dash.vercel.app',
  'http://localhost:3000',
  'http://localhost:5173'
];

// Reserved package names
const RESERVED_NAMES = [
  'pack', 'npm', 'node', 'js', 'python', 'wasm',
  'system', 'admin', 'root', 'config', 'setup',
  'install', 'update', 'remove', 'delete', 'create'
];

// File extension whitelist with expanded support
const ALLOWED_EXTENSIONS = {
  'js': ['js', 'mjs', 'cjs', 'jsx', 'ts', 'tsx'],
  'python': ['py', 'pyc', 'pyo'],
  'wasm': ['wasm'],
  'json': ['json'],
  'markdown': ['md', 'markdown'],
  'text': ['txt'],
  'html': ['html', 'htm'],
```

```
'css': ['css', 'scss', 'sass'],
'image': ['png', 'jpg', 'jpeg', 'gif', 'svg', 'webp'],
'data': ['csv', 'tsv', 'xml', 'yaml', 'yml'],
'binary': ['bin', 'dat']
};
```

```
// ESSENTIAL PACKAGE FILES
```

```
const ESSENTIAL_FILES = [
  'package.json',
  'index.js',
  'main.js',
  'app.js',
  'server.js',
  'README.md',
  'LICENSE',
  'LICENSE.txt',
  'README.txt',
  'CHANGELOG.md',
  'CONTRIBUTING.md'
];
```

```
// ALLOWED NODE.JS MODULES FOR ADVANCED PACKAGES (whitelist)
```

```
const ALLOWED_NODE_MODULES = [
  // Core modules (generally safe)
  'crypto', 'util', 'events', 'stream', 'buffer', 'path', 'url', 'querystring',
  'string_decoder', 'timers', 'console',

  // Common utility modules
  'lodash', 'underscore', 'moment', 'date-fns', 'axios', 'node-fetch',
  'uuid', 'validator', 'joi', 'yup',

  // Data processing
  'csv-parse', 'csv-stringify', 'jsonwebtoken', 'bcrypt', 'bcryptjs',

  // Safe utility modules
  'chalk', 'colors', 'debug', 'winston', 'pino',

  // File formats
  'yaml', 'xml2js', 'cheerio',

  // Testing (safe)
  'jest', 'mocha', 'chai', 'sinon'
];
```

```

// BANNED NODE.JS MODULES (blacklist)
const BANNED_NODE_MODULES = [
  'child_process', 'cluster', 'worker_threads', 'vm',
  'fs', 'os', 'net', 'dns', 'tls', 'http', 'https',
  'dgram', 'zlib', 'perf_hooks', 'repl', 'readline',
  'module', 'process'
];

// Package types with their capabilities
const PACKAGE_TYPES = {
  'basic': {
    level: 1,
    maxFiles: 20,
    maxSize: 5 * 1024 * 1024, // 5MB
    allowNodeModules: false,
    allowAdvancedJS: false,
    requiresVerification: false
  },
  'standard': {
    level: 2,
    maxFiles: 50,
    maxSize: 10 * 1024 * 1024, // 10MB
    allowNodeModules: true,
    allowedNodeModules: ALLOWED_NODE_MODULES.slice(0, 30),
    allowAdvancedJS: true,
    requiresVerification: false
  },
  'advanced': {
    level: 3,
    maxFiles: 100,
    maxSize: 25 * 1024 * 1024, // 25MB
    allowNodeModules: true,
    allowedNodeModules: ALLOWED_NODE_MODULES,
    allowAdvancedJS: true,
    requiresVerification: true,
    sandboxLevel: 'strict'
  }
};

export default async function handler(req, res) {
  // Start timing for performance monitoring
  const startTime = Date.now();

  // Enhanced CORS with stricter headers

```

```

const origin = req.headers.origin;
if (origin && ALLOWED_ORIGINS.includes(origin)) {
  res.setHeader('Access-Control-Allow-Origin', origin);
} else if (origin) {
  console.warn('Unauthorized origin attempt: ${origin}');
}

res.setHeader('Access-Control-Allow-Methods', 'POST, OPTIONS');
res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Origin, X-Requested-With,
Authorization');
res.setHeader('Access-Control-Max-Age', '86400');
res.setHeader('X-Content-Type-Options', 'nosniff');
res.setHeader('X-Frame-Options', 'DENY');
res.setHeader('X-XSS-Protection', '1; mode=block');
res.setHeader('Referrer-Policy', 'strict-origin-when-cross-origin');

// Handle preflight
if (req.method === 'OPTIONS') {
  return res.status(200).end();
}

if (req.method !== 'POST') {
  return res.status(405).json({
    success: false,
    error: 'Method not allowed',
    code: 'METHOD_NOT_ALLOWED'
  });
}

// Enhanced rate limiting with IP banning
const clientIp = req.headers['x-forwarded-for']?.split(',')[0]?.trim() ||
  req.headers['x-real-ip'] ||
  req.socket.remoteAddress;

if (clientIp) {
  const requestData = rateLimitStore.get(clientIp) || {
    count: 0,
    resetTime: Date.now() + RATE_LIMIT_CONFIG.WINDOW_MS,
    violations: 0,
    bannedUntil: 0
  };

  if (requestData.bannedUntil > Date.now()) {
    return res.status(429).json({

```

```

        success: false,
        error: 'IP address temporarily banned due to excessive requests',
        code: 'IP_BANNED',
        retryAfter: Math.ceil((requestData.bannedUntil - Date.now()) / 1000)
    });
}

const now = Date.now();

if (now > requestData.resetTime) {
    requestData.count = 1;
    requestData.resetTime = now + RATE_LIMIT_CONFIG.WINDOW_MS;
} else {
    requestData.count++;

    if (requestData.count > RATE_LIMIT_CONFIG.MAX_REQUESTS) {
        requestData.violations++;

        if (requestData.violations >= RATE_LIMIT_CONFIG.BAN_THRESHOLD) {
            requestData.bannedUntil = now + RATE_LIMIT_CONFIG.BAN_WINDOW_MS;
            console.warn(`IP banned: ${clientIp} for ${RATE_LIMIT_CONFIG.BAN_WINDOW_MS /
1000} seconds`);
        }

        rateLimitStore.set(clientIp, requestData);

        return res.status(429).json({
            success: false,
            error: `Rate limit exceeded. Maximum ${RATE_LIMIT_CONFIG.MAX_REQUESTS}
requests per minute allowed.`,
            code: 'RATE_LIMIT_EXCEEDED',
            retryAfter: Math.ceil((requestData.resetTime - now) / 1000)
        });
    }

    rateLimitStore.set(clientIp, requestData);
}

// Validate Content-Type
const contentType = req.headers['content-type'];
if (!contentType || !contentType.includes('application/json')) {
    return res.status(415).json({
        success: false,

```

```

    error: 'Content-Type must be application/json',
    code: 'INVALID_CONTENT_TYPE'
  });
}

// Size limit check (50MB for advanced packages)
const contentLength = parseInt(req.headers['content-length'] || '0');
if (contentLength > 50 * 1024 * 1024) {
  return res.status(413).json({
    success: false,
    error: 'Request body too large. Maximum 50MB allowed for advanced packages.',
    code: 'PAYLOAD_TOO_LARGE'
  });
}

try {
  const {
    name,
    packJson,
    files,
    isPublic = true,
    packageType = 'basic',
    version = '1.0.0',
    isNewVersion = false,
    basePackId = null,
    userId = null,
    editToken = null,
    collaborators = []
  } = req.body;

  // Validate required fields
  if (!name || !packJson || !files) {
    return res.status(400).json({
      success: false,
      error: 'Missing required fields: name, packJson, and files are required',
      code: 'MISSING_FIELDS'
    });
  }

  // Validate package name
  const nameValidation = validatePackageName(name);
  if (!nameValidation.valid) {
    return res.status(400).json({
      success: false,

```

```

        error: `Invalid package name: ${nameValidation.reason}`,
        code: 'INVALID_PACKAGE_NAME'
    });
}

// Check for reserved names
if (RESERVED_NAMES.includes(name.toLowerCase())) {
    return res.status(400).json({
        success: false,
        error: `Package name "${name}" is reserved and cannot be used`,
        code: 'RESERVED_NAME'
    });
}

// Validate package type
if (!PACKAGE_TYPES[packageType]) {
    return res.status(400).json({
        success: false,
        error: `Invalid package type. Must be one of: ${Object.keys(PACKAGE_TYPES).join(', ')}`,
        code: 'INVALID_PACKAGE_TYPE'
    });
}

const packageConfig = PACKAGE_TYPES[packageType];

// Validate packJson
let packJsonObj;
try {
    packJsonObj = JSON.parse(packJson);

    if (typeof packJsonObj !== 'object' || packJsonObj === null) {
        return res.status(400).json({
            success: false,
            error: 'packJson must be a valid JSON object',
            code: 'INVALID_PACK_JSON'
        });
    }
}

// Validate packJson size
const packJsonSize = JSON.stringify(packJsonObj).length;
if (packJsonSize > 2 * 1024 * 1024) { // 2MB max for advanced packages
    return res.status(400).json({
        success: false,
        error: 'packJson too large. Maximum 2MB allowed.',

```

```

        code: 'PACK_JSON_TOO_LARGE'
    });
}
} catch (e) {
    return res.status(400).json({
        success: false,
        error: 'Invalid packJson: Must be valid JSON',
        code: 'INVALID_JSON'
    });
}

// Validate files object structure
if (typeof files !== 'object' || files === null || Array.isArray(files)) {
    return res.status(400).json({
        success: false,
        error: 'Files must be an object with filename: content pairs',
        code: 'INVALID_FILES_STRUCTURE'
    });
}

// File count limit based on package type
const fileCount = Object.keys(files).length;
if (fileCount > packageConfig.maxFiles) {
    return res.status(400).json({
        success: false,
        error: `Too many files. Maximum ${packageConfig.maxFiles} files allowed for
${packageType} packages.`,
        code: 'TOO_MANY_FILES'
    });
}

if (fileCount === 0) {
    return res.status(400).json({
        success: false,
        error: 'At least one file is required',
        code: 'NO_FILES'
    });
}

// Validate individual files
let totalSize = 0;
const processedFiles = {};
const fileDependencies = new Set();

```

```

for (const [filename, content] of Object.entries(files)) {
  // Validate filename
  const filenameValidation = validateFilename(filename, packageType);
  if (!filenameValidation.valid) {
    return res.status(400).json({
      success: false,
      error: `Invalid filename: ${filenameValidation.reason}`,
      code: 'INVALID_FILENAME'
    });
  }

  // Validate content type and size
  if (typeof content !== 'string') {
    return res.status(400).json({
      success: false,
      error: `File content must be a string: ${filename}`,
      code: 'INVALID_CONTENT_TYPE'
    });
  }

  // Check content size
  const fileSize = content.length;
  totalSize += fileSize;

  // Individual file size limit
  const maxFileSize = packageType === 'advanced' ? 10 * 1024 * 1024 : 2 * 1024 * 1024;
  if (fileSize > maxFileSize) {
    return res.status(400).json({
      success: false,
      error: `File too large: ${filename}. Maximum ${maxFileSize / 1024 / 1024}MB per file.`,
      code: 'FILE_TOO_LARGE'
    });
  }

  // Check for empty files
  if (fileSize === 0 && !filename.endsWith('.json')) {
    return res.status(400).json({
      success: false,
      error: `File cannot be empty: ${filename}`,
      code: 'EMPTY_FILE'
    });
  }

  // File extension validation

```

```

const ext = filename.split('.').pop().toLowerCase();
const fileType = getFileType(ext);

if (!fileType && (!ext || ext === filename)) {
  const essentialNoExt = ESSENTIAL_FILES.filter(f => !f.includes('.'));
  if (!essentialNoExt.includes(filename)) {
    return res.status(400).json({
      success: false,
      error: `Unsupported file extension: ${ext}`,
      code: 'UNSUPPORTED_EXTENSION'
    });
  }
} else if (!fileType) {
  return res.status(400).json({
    success: false,
    error: `Unsupported file extension: ${ext}`,
    code: 'UNSUPPORTED_EXTENSION'
  });
}

// Content validation based on package type
const contentValidation = validateFileContent(filename, content, fileType, packageType);
if (!contentValidation.valid) {
  return res.status(400).json({
    success: false,
    error: `Invalid content in ${filename}: ${contentValidation.reason}`,
    code: 'INVALID_CONTENT'
  });
}

// Extract dependencies from package.json
if (filename.toLowerCase() === 'package.json') {
  try {
    const pkgJson = JSON.parse(content);
    if (pkgJson.dependencies) {
      Object.keys(pkgJson.dependencies).forEach(dep => {
        if (dep.startsWith('@')) return; // Skip scoped packages
        fileDependencies.add(dep.toLowerCase());
      });
    }
  } catch (e) {
    // Ignore parsing errors, will be caught by validation
  }
}

```

```

    // Store sanitized content
    processedFiles[filename] = content;
}

// Total package size limit
if (totalSize > packageConfig.maxSize) {
    return res.status(400).json({
        success: false,
        error: `Package too large. Total size: ${totalSize / 1024 / 1024}.toFixed(2)}MB. Maximum
        ${packageConfig.maxSize / 1024 / 1024}MB for ${packageType} packages.`,
        code: 'PACKAGE_TOO_LARGE'
    });
}

// Validate dependencies for advanced packages
if (packageType === 'advanced' || packageType === 'standard') {
    for (const dep of fileDependencies) {
        const allowed = packageConfig.allowedNodeModules || ALLOWED_NODE_MODULES;
        if (!allowed.includes(dep) && !allowed.some(a => dep.startsWith(a + '/')) {
            return res.status(400).json({
                success: false,
                error: `Dependency "${dep}" is not allowed for ${packageType} packages.`,
                code: 'DISALLOWED_DEPENDENCY',
                allowedModules: allowed
            });
        }
    }

    // Check banned modules
    if (BANNED_NODE_MODULES.includes(dep)) {
        return res.status(400).json({
            success: false,
            error: `Dependency "${dep}" is banned for security reasons.`,
            code: 'BANNED_DEPENDENCY'
        });
    }
}
}

// VERSIONING AND COLLABORATION LOGIC
let versionNumber = sanitizeVersion(version);

if (isNewVersion && basePackId) {
    // Create new version of existing pack

```

```

const { data: basePack, error: baseError } = await supabase
  .from('packs')
  .select('id, name, version, publisher_id')
  .eq('id', basePackId)
  .single();

if (baseError || !basePack) {
  return res.status(404).json({
    success: false,
    error: 'Base package not found',
    code: 'BASE_PACK_NOT_FOUND'
  });
}

// Check if user can edit this pack using new table
const canEdit = await canUserEditPack(basePackId, userId, editToken);
if (!canEdit) {
  return res.status(403).json({
    success: false,
    error: 'You do not have permission to edit this package',
    code: 'EDIT_PERMISSION_DENIED',
    suggestion: 'Request edit access from the pack owner or use an edit token'
  });
}

// Check if version already exists
const { data: existingVersion } = await supabase
  .from('pack_versions')
  .select('version')
  .eq('pack_id', basePackId)
  .eq('version', versionNumber)
  .single();

if (existingVersion) {
  return res.status(409).json({
    success: false,
    error: `Version ${versionNumber} already exists for this package`,
    code: 'VERSION_EXISTS',
    suggestion: 'Use a different version number'
  });
}
} else {
  // New package - check for existing name
  const { data: existingPack, error: checkError } = await supabase

```

```

    .from('packs')
    .select('id, name, created_at')
    .eq('name', name)
    .limit(1);

    if (checkError) {
      console.error('Duplicate check error:', checkError);
      return res.status(500).json({
        success: false,
        error: 'Internal server error during duplicate check',
        code: 'DUPLICATE_CHECK_FAILED'
      });
    }

    if (existingPack && existingPack.length > 0) {
      const existing = existingPack[0];
      const createdAt = new Date(existing.created_at).toLocaleDateString();

      return res.status(409).json({
        success: false,
        error: `Package name "${name}" is already taken. Package names must be unique.`,
        code: 'PACKAGE_NAME_EXISTS',
        existingPackageId: existing.id,
        existingPackageCreated: createdAt,
        suggestion: `Use "isNewVersion: true" and "basePackId: "${existing.id}" to create a new
version`
      });
    }
  }

  // Generate cryptographically secure URL ID
  const urlId = generateSecureUrlId();

  // Verify URL ID uniqueness
  const { data: existingUrlId } = await supabase
    .from('packs')
    .select('id')
    .eq('url_id', urlId)
    .limit(1);

  if (existingUrlId && existingUrlId.length > 0) {
    return res.status(500).json({
      success: false,
      error: 'Internal server error: URL ID collision',
    });
  }

```

```

        code: 'URL_ID_COLLISION'
    });
}

// Generate URLs
const cdnUrl = `https://packcdn.firefly-worker.workers.dev/cdn/${urlId}`;
const workerUrl = `https://packcdn.firefly-worker.workers.dev/pack/${urlId}`;

// Generate encryption key for private packages
const encryptedKey = !isPublic ? generateSecureEncryptionKey() : null;

// Generate checksum
const packageChecksum = generateChecksum(JSON.stringify(processedFiles));

// Save to main packs table (using existing columns)
const now = new Date().toISOString();
const packData = {
    url_id: urlId,
    name,
    pack_json: packJson,
    files: processedFiles,
    cdn_url: cdnUrl,
    worker_url: workerUrl,
    encrypted_key: encryptedKey,
    is_public: isPublic,
    version: versionNumber,
    created_at: now,
    updated_at: now,
    views: 0,
    downloads: 0,
    publish_ip: clientIp,
    last_accessed: now,
    publisher_id: userId
};

// Insert into main packs table
const { data: pack, error } = await supabase
    .from('packs')
    .insert([packData])
    .select()
    .single();

if (error) {
    console.error('Supabase insert error:', error);
}

```

```

if (error.code === '23505') {
  return res.status(409).json({
    success: false,
    error: 'Package with this name already exists',
    code: 'DUPLICATE_PACKAGE'
  });
}

console.error('Database error details:', {
  code: error.code,
  message: error.message,
  details: error.details,
  hint: error.hint
});

return res.status(500).json({
  success: false,
  error: 'Failed to save package. Please try again.',
  code: 'DATABASE_ERROR'
});
}

// Save to new tables for advanced features
try {
  // 1. Save to pack_versions table
  const { data: versionData } = await supabase
    .from('pack_versions')
    .insert([
      {
        pack_id: pack.id,
        version: versionNumber,
        version_number: isNewVersion ? await getNextVersionNumber(basePackId) : 1,
        pack_json: packJson,
        files: processedFiles,
        checksum: packageChecksum,
        publisher_id: userId,
        created_at: now,
        updated_at: now
      }
    ])
    .select()
    .single();

  // 2. Save to pack_metadata table
  await supabase

```

```

    .from('pack_metadata')
    .insert([
      {
        pack_id: pack.id,
        package_type: packageType,
        sandbox_level: packageConfig.sandboxLevel || 'basic',
        requires_verification: packageConfig.requiresVerification || false,
        verification_status: packageConfig.requiresVerification ? 'pending' : 'approved',
        file_count: fileCount,
        total_size: totalSize,
        last_accessed: now,
        updated_at: now
      }
    ]);

// 3. Save dependencies to pack_dependencies table
if (fileDependencies.size > 0) {
  const dependencyInserts = Array.from(fileDependencies).map(dep => ({
    pack_id: pack.id,
    dependency_name: dep,
    created_at: now
  }));

  await supabase
    .from('pack_dependencies')
    .insert(dependencyInserts);
}

// 4. Save collaborators to pack_collaborators table
if (collaborators && Array.isArray(collaborators)) {
  const validCollaborators = collaborators.filter(c =>
    c && typeof c === 'string' && c.length > 0
  ).slice(0, 10);

  if (validCollaborators.length > 0) {
    // Add current user as admin if not already in list
    if (userId && !validCollaborators.includes(userId)) {
      validCollaborators.unshift(userId);
    }

    const collaboratorInserts = validCollaborators.map((collabUserId, index) => ({
      pack_id: pack.id,
      user_id: collabUserId,
      permission_level: index === 0 ? 'admin' : 'editor',
      invited_by: userId,
      accepted_at: now,
    }));
  }
}

```

```

        created_at: now
    }));

    await supabase
      .from('pack_collaborators')
      .insert(collaboratorInserts);
  }
}

// 5. Log to pack_changes table
await supabase
  .from('pack_changes')
  .insert([
    {
      pack_id: pack.id,
      user_id: userId,
      change_type: isNewVersion ? 'version' : 'create',
      description: isNewVersion
        ? `Created new version ${versionNumber} from base pack ${basePackId}`
        : `Created new package ${name} v${versionNumber}`,
      metadata: {
        packageType,
        fileCount,
        totalSize,
        isPublic,
        isNewVersion,
        basePackId
      },
      created_at: now
    }
  ]);

} catch (advancedError) {
  console.warn('Advanced features save failed:', advancedError);
  // Don't fail the entire publish if advanced features fail
}

// Log successful publish
const processingTime = Date.now() - startTime;
console.log(`Package published successfully:`, {
  name,
  urlId,
  packageType,
  version: versionNumber,
  fileCount,
  totalSize: `${(totalSize / 1024).toFixed(2)}KB`,

```

```

    processingTime: `${processingTime}ms`,
    clientIp,
    isNewVersion,
    basePackId
  });

  // Return success response
  res.status(201).json({
    success: true,
    packId: pack.id,
    urlId: urlId,
    cdnUrl,
    workerUrl,
    installCommand: `pack install ${name}@${versionNumber} ${cdnUrl}`,
    encryptedKey,
    isNewVersion,
    basePackId,
    version: versionNumber,
    metadata: {
      name,
      version: versionNumber,
      packageType,
      fileCount,
      totalSize,
      isPublic,
      dependencies: Array.from(fileDependencies),
      createdAt: now,
      checksum: packageChecksum
    },
    links: {
      cdn: cdnUrl,
      info: workerUrl,
      download: `${cdnUrl}/index.js`,
      api: `/api/get-pack?id=${urlId}`,
      versions: `/api/pack-versions?id=${pack.id}`
    },
    advancedFeatures: {
      versioning: true,
      collaboration: true,
      dependencies: fileDependencies.size > 0,
      metadata: true
    }
  });

```

```

} catch (error) {
  console.error('Publish error:', {
    message: error.message,
    stack: error.stack,
    clientIp,
    timestamp: new Date().toISOString(),
    processingTime: `${Date.now() - startTime}ms`
  });

  res.status(500).json({
    success: false,
    error: 'An unexpected error occurred. Please try again later.',
    code: 'INTERNAL_SERVER_ERROR'
  });
}
}

```

// NEW HELPER FUNCTIONS FOR ADVANCED FEATURES

```

async function canUserEditPack(packId, userId, editToken = null) {
  // Public editing - anyone can create new versions
  if (!userId) {
    return true;
  }

```

```

  // Check if user is in pack_collaborators table

```

```

  const { data: collaborator } = await supabase
    .from('pack_collaborators')
    .select('permission_level, accepted_at')
    .eq('pack_id', packId)
    .eq('user_id', userId)
    .single();

```

```

  if (collaborator && collaborator.accepted_at &&
    ['editor', 'admin'].includes(collaborator.permission_level)) {
    return true;
  }

```

```

  // Check if user is the original publisher

```

```

  const { data: pack } = await supabase
    .from('packs')
    .select('publisher_id')
    .eq('id', packId)
    .single();

```

```

if (pack && pack.publisher_id === userId) {
  return true;
}

// Check edit token if provided
if (editToken) {
  const { data: token } = await supabase
    .from('edit_tokens')
    .select('expires_at, max_uses, use_count')
    .eq('token', editToken)
    .eq('pack_id', packId)
    .single();

  if (token) {
    const now = new Date();
    const expiresAt = new Date(token.expires_at);

    if (expiresAt > now &&
      (token.max_uses === 0 || token.use_count < token.max_uses)) {

      // Increment use count
      await supabase
        .from('edit_tokens')
        .update({ use_count: token.use_count + 1 })
        .eq('token', editToken);

      return true;
    }
  }
}

return false;
}

async function getNextVersionNumber(packId) {
  const { data: versions } = await supabase
    .from('pack_versions')
    .select('version_number')
    .eq('pack_id', packId)
    .order('version_number', { ascending: false })
    .limit(1);

  return versions && versions.length > 0 ? versions[0].version_number + 1 : 1;
}

```

```
}
```

```
// EXISTING HELPER FUNCTIONS (unchanged)
```

```
async function verifyEditPermission(packId, userId, editToken, collaborators = []) {  
  return await canUserEditPack(packId, userId, editToken);  
}
```

```
function validatePackageName(name) {  
  if (typeof name !== 'string') {  
    return { valid: false, reason: 'Package name must be a string' };  
  }
```

```
  if (name.length < 2) {  
    return { valid: false, reason: 'Package name must be at least 2 characters' };  
  }
```

```
  if (name.length > 50) {  
    return { valid: false, reason: 'Package name cannot exceed 50 characters' };  
  }
```

```
  if (!/^[a-z]/.test(name)) {  
    return { valid: false, reason: 'Package name must start with a lowercase letter' };  
  }
```

```
  if (!/^[a-z0-9-_.]+$/.test(name)) {  
    return { valid: false, reason: 'Package name can only contain lowercase letters, numbers,  
hyphens, and underscores' };  
  }
```

```
  if (/[-_]$/ .test(name)) {  
    return { valid: false, reason: 'Package name cannot end with a hyphen or underscore' };  
  }
```

```
  if (/[-_]{2,}/.test(name)) {  
    return { valid: false, reason: 'Package name cannot contain consecutive hyphens or  
underscores' };  
  }
```

```
  const offensivePatterns = [  
    /admin/i, /root/i, /system/i, /config/i,  
    /password/i, /token/i, /secret/i, /private/i  
  ];
```

```

    if (offensivePatterns.some(pattern => pattern.test(name))) {
      return { valid: false, reason: 'Package name contains restricted patterns' };
    }

    return { valid: true };
  }

function validateFilename(filename, packageType) {
  if (typeof filename !== 'string') {
    return { valid: false, reason: 'Filename must be a string' };
  }

  if (filename.length === 0) {
    return { valid: false, reason: 'Filename cannot be empty' };
  }

  if (filename.length > 255) {
    return { valid: false, reason: 'Filename cannot exceed 255 characters' };
  }

  if (!/^[a-zA-Z0-9._\-\+$/].test(filename)) {
    return { valid: false, reason: 'Filename can only contain letters, numbers, dots, hyphens, and underscores' };
  }

  if (filename.includes('.') || filename.includes('/') || filename.includes("\\")) {
    return { valid: false, reason: 'Filename cannot contain directory traversal characters' };
  }

  const allowedHiddenFiles = [
    '.gitignore', '.env', '.npmrc', '.prettierrc', '.eslintrc',
    '.babelrc', '.dockerignore', '.gitattributes', '.editorconfig'
  ];

  if (filename.startsWith('.') && !allowedHiddenFiles.some(allowed =>
    filename === allowed || filename.startsWith(allowed + '/'))) {
    return { valid: false, reason: 'Hidden files are only allowed for common configuration files' };
  }

  const essentialFileLower = filename.toLowerCase();
  if (ESSENTIAL_FILES.some(essential => essentialFileLower === essential.toLowerCase())) {
    return { valid: true };
  }
}

```

```

const reservedDirectories = [
  'node_modules', 'vendor', 'lib', 'bin', 'dist', 'build',
  'public', 'src', 'test', 'tests', 'docs', 'examples'
];

if (reservedDirectories.includes(filename.toLowerCase())) {
  return { valid: false, reason: 'Filename is reserved for system use' };
}

const ext = filename.split('.').pop().toLowerCase();
if (!ext || ext === filename) {
  const allowedNoExt = /^[A-Z][A-Z0-9_]*(\.[A-Z0-9]+)?$/;
  if (allowedNoExt.test(filename)) {
    return { valid: true };
  }
  return { valid: false, reason: 'Files without extensions must follow common naming conventions' };
}

// Advanced packages can have more file types
if (packageType === 'basic' && ['ts', 'tsx', 'scss', 'sass'].includes(ext)) {
  return { valid: false, reason: `File extension .${ext} requires standard or advanced package type` };
}

return { valid: true };
}

function getFileType(extension) {
  for (const [type, exts] of Object.entries(ALLOWED_EXTENSIONS)) {
    if (exts.includes(extension)) {
      return type;
    }
  }
  return null;
}

function validateFileContent(filename, content, fileType, packageType) {
  if (/\\x00/.test(content)) {
    return { valid: false, reason: 'File contains null bytes' };
  }

  const lines = content.split('\\n');
  for (const line of lines) {

```

```

    if (line.length > 10000) {
      return { valid: false, reason: 'File contains excessively long lines' };
    }
  }
}

const filenameLower = filename.toLowerCase();

if (ESSENTIAL_FILES.some(essential => filenameLower === essential.toLowerCase())) {
  if (filenameLower === 'package.json' || filename.endsWith('.json')) {
    try {
      JSON.parse(content);
      return { valid: true };
    } catch (e) {
      return { valid: false, reason: 'Invalid JSON format in package.json' };
    }
  }
  return { valid: true };
}

switch (fileType) {
  case 'js':
    return validateJavaScript(content, packageType);
  case 'json':
    return validateJSON(content);
  default:
    return { valid: true };
}
}

function validateJavaScript(content, packageType) {
  const dangerousPatterns = [
    /\beval\s*\(/i,
    /\bFunction\s*\(/i,
    /\bnew\s+Function\s*\(/i,
    /\bsetTimeout\s*\([^)]*\)/i,
    /\bsetInterval\s*\([^)]*\)/i,
    /\bsetImmediate\s*\([^)]*\)/i
  ];

  // Allow more patterns for advanced packages
  if (packageType === 'basic') {
    dangerousPatterns.push(
      /\brequire\s*\([^)]*\)/i,
      /\bprocess\s*\./i,

```

```

    /\bfs\s*\./i,
    /\bfetch\s*\([^)]*\)/i,
    /\bXMLHttpRequest/i
  );
}

for (const pattern of dangerousPatterns) {
  if (pattern.test(content)) {
    const reason = packageType === 'basic'
      ? 'Basic packages cannot use dynamic imports or I/O operations'
      : 'JavaScript contains potentially dangerous patterns';
    return { valid: false, reason };
  }
}

const avgLineLength = content.length / (content.split('\n').length || 1);
if (avgLineLength > 1000) {
  return { valid: false, reason: 'JavaScript appears to be minified or obfuscated' };
}

return { valid: true };
}

function validateJSON(content) {
  try {
    JSON.parse(content);
    return { valid: true };
  } catch (e) {
    return { valid: false, reason: 'Invalid JSON format' };
  }
}

function generateSecureUrlId() {
  return crypto.randomBytes(12).toString('hex') +
    Date.now().toString(36) +
    crypto.randomBytes(4).toString('hex');
}

function generateSecureEncryptionKey() {
  return crypto.randomBytes(32).toString('hex');
}

function sanitizeVersion(version) {
  if (typeof version !== 'string') return '1.0.0';

```

```

// Remove everything except numbers, dots, and prerelease identifiers
const sanitized = version.replace(/[^0-9.a-zA-Z-+]/g, "");

if (!sanitized.includes('.')) {
  return sanitized + '.0.0';
}

const parts = sanitized.split('.').filter(part => part !== "");
if (parts.length === 0) return '1.0.0';

const numericParts = parts.map(part => {
  const numMatch = part.match(/^(d+)/);
  return numMatch ? numMatch[1] : '0';
});

while (numericParts.length < 3) {
  numericParts.push('0');
}

return numericParts.slice(0, 5).join('.');
}

function generateChecksum(data) {
  return crypto.createHash('sha256').update(data).digest('hex');
}

// Cleanup rate limiting store
setInterval(() => {
  const now = Date.now();
  for (const [ip, data] of rateLimitStore.entries()) {
    if (now > data.resetTime + 60 * 60 * 1000 && data.bannedUntil < now) {
      rateLimitStore.delete(ip);
    }
  }
}, 5 * 60 * 1000);

// Export for testing
if (process.env.NODE_ENV === 'test') {
  module.exports = {
    validatePackageName,
    validateFilename,
    validateFileContent,
    sanitizeVersion,
  };
}

```

```

    generateSecureUrlId,
    PACKAGE_TYPES,
    ALLOWED_NODE_MODULES,
    BANNED_NODE_MODULES,
    canUserEditPack,
    getNextVersionNumber
  };
}

```

[Random-URLS.js](#)

```

// api/random-urls.js
export default function handler(req, res) {
  const { action, count = 100, seed } = req.query;

  // Your actual files
  const targets = [
    { id: 'home', path: '/selector.html', name: 'Home Selector' },
    { id: 'dashboard', path: '/home/index.html', name: 'Dashboard' },
    { id: 'docs', path: '/docs.html', name: 'Documentation' },
    { id: 'editor', path: '/editor.html', name: 'Editor' },
    { id: 'explore', path: '/explore.html', name: 'Explorer' },
    { id: 'config', path: '/config.json', name: 'Configuration' }
  ];

  // Massive word banks for infinite combinations
  const wordBanks = {
    tech: ['quantum', 'cyber', 'digital', 'virtual', 'neural', 'synth', 'crypto', 'blockchain', 'ai', 'ml'],
    space: ['cosmic', 'stellar', 'galactic', 'orbital', 'lunar', 'solar', 'nebula', 'pulsar', 'quasar',
'wormhole'],
    nature: ['forest', 'ocean', 'mountain', 'river', 'crystal', 'ember', 'blaze', 'frost', 'storm', 'thunder'],
    fantasy: ['dragon', 'phoenix', 'wizard', 'arcane', 'mythic', 'legend', 'rune', 'spell', 'enchanted',
'magic'],
    future: ['neo', 'ultra', 'hyper', 'mega', 'tera', 'peta', 'omega', 'alpha', 'beta', 'gamma'],
    portals: ['gate', 'portal', 'door', 'window', 'bridge', 'tunnel', 'path', 'route', 'gateway', 'access'],
    places: ['nexus', 'hub', 'core', 'center', 'matrix', 'grid', 'network', 'web', 'cloud', 'cluster']
  };

  // Generate a random URL path
  function generateRandomUrl(targetId, index) {
    const categories = Object.keys(wordBanks);
    const numWords = 2 + Math.floor(Math.random() * 3); // 2-4 words

    let path = '/';

```

```

for (let i = 0; i < numWords; i++) {
  const category = categories[Math.floor(Math.random() * categories.length)];
  const words = wordBanks[category];
  const word = words[Math.floor(Math.random() * words.length)];
  path += word + (i < numWords - 1 ? '-' : '');
}

// Add some variations
const variations = [
  () => path + '-v' + (Math.floor(Math.random() * 10) + 1),
  () => path + Math.floor(Math.random() * 1000),
  () => path + '-' + Date.now().toString(36),
  () => '/api/' + path.slice(1),
  () => '/v' + (Math.floor(Math.random() * 3) + 1) + path,
  () => path
];

return variations[Math.floor(Math.random() * variations.length)]();
}

// Generate hundreds of URLs
if (action === 'generate' || !action) {
  const urlCount = Math.min(parseInt(count), 1000); // Cap at 1000
  const generatedUrls = [];

  for (let i = 0; i < urlCount; i++) {
    const target = targets[Math.floor(Math.random() * targets.length)];
    const randomUrl = generateRandomUrl(target.id, i);

    // CORRECTED: Removed duplicate domain in shortUrl
    generatedUrls.push({
      url: randomUrl,
      target: target.path,
      name: target.name,
      id: target.id,
      qrCode:
`https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=${encodeURIComponent('http
s://pack-cdn.vercel.app' + randomUrl)}`,
      shortUrl: `https://go.pack-cdn.vercel.app/${Date.now().toString(36)}${i.toString(36)}`,
      metadata: {
        generated: new Date().toISOString(),
        index: i,
        seed: seed || 'random'
      }
    })
  }
}

```

```

    });
  }

  res.setHeader('Content-Type', 'application/json');
  res.setHeader('Access-Control-Allow-Origin', '*');

  return res.status(200).json({
    success: true,
    count: generatedUrls.length,
    generated: new Date().toISOString(),
    seed: seed || Math.random().toString(36).substring(2),
    urls: generatedUrls,
    endpoints: {
      all: '/api/random-urls',
      generate: '/api/random-urls?action=generate&count=100',
      redirect: '/api/random-urls/redirect/{any-random-path}',
      discover: '/api/random-urls?action=discover'
    },
    usage: 'Use any generated URL - they all redirect to actual pages!'
  });
}

// Discovery mode - find random URLs
if (action === 'discover') {
  const discoveries = [];
  const discoveryCount = Math.min(parseInt(count) || 20, 50);

  for (let i = 0; i < discoveryCount; i++) {
    const target = targets[Math.floor(Math.random() * targets.length)];
    const randomUrl = generateRandomUrl(target.id, i + 1000);

    discoveries.push({
      discoveredUrl: randomUrl,
      leadsTo: target.name,
      probability: Math.random().toFixed(2),
      hint: `Try ${randomUrl} to access ${target.name}`,
      timestamp: new Date(Date.now() + i * 1000).toISOString()
    });
  }

  return res.json({
    action: 'discovery',
    message: 'Found these random endpoints in the wild!',
    discoveries,
  });
}

```

```

    totalDiscovered: discoveryCount,
    nextDiscovery:
`/api/random-urls?action=discover&count=${discoveryCount}&seed=${Date.now()}`
  });
}

// Default: redirect handler
const requestedPath = req.url.replace('/api/random-urls', '');

if (requestedPath && requestedPath !== '/') {
  // Any random path gets redirected to a random target
  const target = targets[Math.floor(Math.random() * targets.length)];

  // Log the random access
  console.log(`Random access: ${requestedPath} -> ${target.path} at ${new
Date().toISOString()}`);

  return res.redirect(302, target.path);
}

// Show API info
res.json({
  name: 'Infinite Random URL Generator',
  description: 'Generates hundreds of random URLs that redirect to your actual pages',
  version: '1.0.0',
  endpoints: {
    generate: '/api/random-urls?action=generate&count=100',
    discover: '/api/random-urls?action=discover',
    redirect: 'Any path under /api/random-urls/ will redirect randomly',
    example: '/api/random-urls/quantum-dragon-nexus'
  },
  features: [
    'Auto-generates 100s of random URLs',
    'Every URL redirects to actual content',
    'Self-discovering endpoint system',
    'QR code generation for each URL',
    'Deterministic but random-looking paths'
  ]
});
}

```

[Wildcard-redirect.js](#)

// /api/wildcard-redirect.js - FIXED & ENHANCED VERSION

```
import { createClient } from '@supabase/supabase-js'

// Initialize Supabase
const supabaseUrl = process.env.SUPABASE_URL
const supabaseKey = process.env.SUPABASE_SERVICE_KEY
const supabase = createClient(supabaseUrl, supabaseKey)

export default async function handler(req, res) {
  const fullPath = req.url;
  const path = fullPath.split('?')[0];
  const queryParams = new URLSearchParams(fullPath.split('?')[1] || "");

  console.log(`🔗 Processing: ${path}`);

  // All your original pages
  const allPages = [
    '/config.json',
    '/docs.html',
    '/Docs/docs.html',
    '/Docs/Pages/Home/index.html',
    '/Docs/Pages/Home/Pages/Editor/editor.html',
    '/Docs/Pages/Home/Pages/Explore/explore.html',
    '/editor.html',
    '/Editor/editor.html',
    '/explore.html',
    '/Explore/explore.html',
    '/home/editor.html',
    '/home/explore.html',
    '/home/index.html',
    '/Login/login.html',
    '/Private/admin.html',
    '/selector.html',
    '/embed.html',
    '/dev-panel.html'
  ];

  // Reserved @names for system pages (NO SLASH)
  const reservedNames = [
    'quantum', 'cosmic', 'digital', 'neo', 'virtual', 'synth', 'stellar',
    'cyber', 'orbital', 'dragon', 'phoenix', 'homecore', 'editornetwork',
    'explorervortex', 'homestudio', 'homediscovery', 'random', 'embed'
  ];

  // --- FIX: Handle @random correctly ---
```

```

// Extract @username from path
let username = null;
if (path.startsWith('/@')) {
  username = path.substring(2).split('?')[0].split('/')[0];
} else if (path.startsWith('@') && path.length > 1) {
  // Handle direct @username (no slash)
  username = path.substring(1).split('?')[0].split('/')[0];
}

// SPECIAL CASE: @random should go directly to dev-panel
if (username === 'random') {
  console.log(`🎨 Dev Panel: ${path} → /dev-panel.html`);
  return res.redirect(302, '/dev-panel.html');
}



// Handle @embed - Embedded websites
if (username === 'embed' && path.includes('/')) {
  const embedPath = path.includes('@embed/')
    ? path.split('@embed/')[1]
    : path.split('embed/')[1];
  if (embedPath) {
    const embedUrl = decodeURIComponent(embedPath);
    console.log(`🌐 Embed Request: ${path} → /embed.html?url=${embedUrl}`);
    return res.redirect(302, `/embed.html?url=${encodeURIComponent(embedUrl)}`);
  }
}



// --- Handle user pages from Supabase ---
if (username && !reservedNames.includes(username)) {
  try {
    // Fetch page from Supabase
    const { data: page, error } = await supabase
      .from('user_pages')
      .select('*')
      .eq('page_id', username)
      .eq('is_public', true)
      .maybeSingle();

    if (error) {
      console.log(`❌ Supabase query error for @${username}:`, error.message);
      // Fall through
    } else if (page) {
      console.log(`📄 User Page Found: @${username} - "${page.title}" (Type: ${page.page_type})`);
    }
  }
}

```

```

// Increment view count
supabase
  .from('user_pages')
  .update({ views: (page.views || 0) + 1 })
  .eq('id', page.id)
  .then(() => console.log( View count updated for @${username}`))
  .catch(e => console.log( View count update failed:`, e.message));

// Render the page based on type
return res.send(renderUserPage(username, page));
} else {
  console.log( No page found for @${username}, falling back to original logic`);
  // Continue to original logic
}
} catch (error) {
  console.log( Error loading user page @${username}:`, error.message);
  // Fall through
}
}

```

// --- ORIGINAL LOGIC (for backward compatibility) ---

// MASSIVE 100+ WORD BANKS FOR ENCRYPTED CODES

```

const wordBanks = {
  tech: [
    'quantum', 'cyber', 'digital', 'virtual', 'neural', 'synth', 'crypto', 'blockchain',
    'ai', 'machine', 'learning', 'bot', 'data', 'cloud', 'server', 'client', 'network',
    'protocol', 'binary', 'byte', 'pixel', 'render', 'stream', 'buffer', 'cache',
    'memory', 'storage', 'database', 'api', 'sdk', 'framework', 'library', 'module',
    'interface', 'terminal', 'console', 'shell', 'kernel', 'driver', 'firmware', 'hardware'
  ],
  space: [
    'cosmic', 'stellar', 'galactic', 'orbital', 'lunar', 'solar', 'nebula', 'pulsar',
    'quasar', 'wormhole', 'blackhole', 'singularity', 'eventhorizon', 'supernova',
    'constellation', 'galaxy', 'universe', 'multiverse', 'dimension', 'reality'
  ],
  fantasy: [
    'dragon', 'phoenix', 'wizard', 'mage', 'sorcerer', 'warlock', 'witch', 'arcane',
    'mythic', 'legend', 'rune', 'spell', 'enchanted', 'magic', 'magical', 'knight'
  ],
  nature: [
    'forest', 'jungle', 'rainforest', 'woodland', 'grove', 'thicket', 'copse', 'orchard',
    'vineyard', 'farm', 'field', 'meadow', 'pasture', 'prairie', 'savanna', 'steppe'
  ]
}

```

```

    ]
};

const allWords = [
  ...wordBanks.tech,
  ...wordBanks.space,
  ...wordBanks.fantasy,
  ...wordBanks.nature
];

// Hash function
function getHash(str) {
  let hash = 0;
  for (let i = 0; i < str.length; i++) {
    hash = ((hash << 5) - hash) + str.charCodeAt(i);
    hash = hash & hash;
  }
  return Math.abs(hash);
}

// Generate encrypted code
function generateEncryptedCode(input) {
  const hash = getHash(input);
  const word1 = allWords[(hash * 1) % allWords.length];
  const word2 = allWords[(hash * 3) % allWords.length];
  const number = (hash % 9999).toString().padStart(4, '0');

  const formats = [
    `@${word1}-${word2}-${number}`,
    `@${word1.slice(0, 3)}-${word2.slice(0, 3)}-${number}`,
    `@${word1}-${number}-${word2}`,
    `@${word1.slice(0, 4)}${word2.slice(0, 3)}${number.slice(0, 3)}`
  ];

  return formats[hash % formats.length].toLowerCase();
}

// Generate fun @username
function generateFunUsername(input) {
  const code = generateEncryptedCode(input + Date.now() + Math.random());
  return `/${code}`;
}

// In-memory storage

```

```


const urlMappings = new Map();

// Initialize with some sample mappings
for (let i = 0; i < Math.min(20, allPages.length); i++) {
  const funUrl = generateFunUsername(`init${i}${allPages[i]}`);
  urlMappings.set(funUrl, allPages[i]);
}


// Handle reserved names redirect
if (username && reservedNames.includes(username) && !path.startsWith('/@')) {
  return res.redirect(301, `/${@username}${path.substring(username.length + 1)}`);
}

// Handle /@ paths for backward compatibility
if (path.startsWith('/@')) {
  const pathUsername = path.substring(2).split('?')[0].split('/')[0];

  if (!reservedNames.includes(pathUsername)) {
    // Check Supabase for user page
    try {
      const { data: page } = await supabase
        .from('user_pages')
        .select('*')
        .eq('page_id', pathUsername)
        .eq('is_public', true)
        .maybeSingle();

      if (page) {
        // Redirect to @username format (no slash)
        return res.redirect(301, `/${@pathUsername}`);
      }
    } catch (error) {
      console.log(` No Supabase page for /@${pathUsername}:`, error.message);
    }
  }
}

// Check if it's a file path that should get a fun URL
if (allPages.includes(path)) {
  const funUrl = generateFunUsername(path + Date.now());
  urlMappings.set(funUrl, path);

  console.log(` File: ${path} → ${funUrl}`);
}

```

```

    res.setHeader('Cache-Control', 'no-store');
    res.setHeader('X-Encrypted-URL', funUrl);
    res.setHeader('X-Original-File', path);

    return res.redirect(301, funUrl);
  }

  // Handle @ paths - ANY @anything gets mapped
  if (path.startsWith('/@')) {
    if (urlMappings.has(path)) {
      const targetFile = urlMappings.get(path);
      console.log(`🔑 Known @path: ${path} → ${targetFile}`);

      res.setHeader('Cache-Control', 'no-store');
      res.setHeader('X-Target-File', targetFile);

      return res.redirect(302, targetFile);
    }

    const randomIndex = Math.floor(Math.random() * allPages.length);
    const targetFile = allPages[randomIndex];

    urlMappings.set(path, targetFile);
    console.log(`🌟 NEW @path: ${path} → ${targetFile}`);

    res.setHeader('Cache-Control', 'no-store');
    res.setHeader('X-New-Mapping', 'true');
    res.setHeader('X-Target-File', targetFile);

    return res.redirect(302, targetFile);
  }

  // Handle direct requests to embed.html
  if (path === '/embed.html' && queryParams.has('url')) {
    const embedUrl = queryParams.get('url');
    console.log(`🌐 Direct Embed: ${path}?url=${embedUrl}`);

    return res.send(`
      <!DOCTYPE html>
      <html>
        <head>
          <meta http-equiv="refresh"
content="0;url=/${@embed/${encodeURIComponent(embedUrl)}}">
        </head>
    `);
  }

```

```

    <body>
      <p>Redirecting to embedded page...</p>
    </body>
  </html>
`);
}

// Show system info
if (path === '/url-system' || path === '/info') {
  let pageCount = 0;
  let topPages = [];

  try {
    const { count, error } = await supabase
      .from('user_pages')
      .select('*', { count: 'exact', head: true });

    if (!error) pageCount = count || 0;

    const { data, error: topError } = await supabase
      .from('user_pages')
      .select('page_id, title, views, created_at')
      .eq('is_public', true)
      .order('views', { ascending: false })
      .limit(10);

    if (!topError) topPages = data || [];
  } catch (error) {
    console.log('Supabase stats error:', error.message);
  }

  const samples = [];
  for (let i = 0; i < 10; i++) {
    samples.push(generateFunUsername(`sample${i}`));
  }

  return res.json({
    system: 'Pack CDN URL System v3.1',
    status: 'active',
    features: [
      '@username - User pages (Supabase)',
      '@random - Developer Panel',
      '@embed/* - Embedded websites',
      'Encrypted URLs',
    ],
  });
}

```

```

    'HTML Content Rendering'
  ],
  stats: {
    userPages: pageCount,
    systemPages: allPages.length,
    topPages: topPages,
    totalWords: allWords.length,
    totalMappings: urlMappings.size
  },
  sampleUrls: samples,
  usage: [
    'pack-cdn.vercel.app@username - Access user page',
    'pack-cdn.vercel.app/@username - Alternative access',
    'pack-cdn.vercel.app@random - Create pages',
    'pack-cdn.vercel.app@embed/url - Embed websites'
  ]
});
}

```

```

// For any other path, generate new encrypted URL
const funUrl = generateFunUsername(path + Date.now());
const randomIndex = Math.floor(Math.random() * allPages.length);
const targetFile = allPages[randomIndex];

```

```

urlMappings.set(funUrl, targetFile);

```

```

console.log(`🚀 Random path: ${path} → ${funUrl} → ${targetFile}`);

```

```

res.setHeader('Cache-Control', 'no-store');
res.setHeader('X-Generated-URL', funUrl);
res.setHeader('X-Target-File', targetFile);

```

```

return res.redirect(302, funUrl);
}

```

```

// ENHANCED renderUserPage with proper HTML handling

```

```

function renderUserPage(username, pageData) {
  console.log(`💻 Rendering page @${username} of type: ${pageData.page_type}`);

```

```

  // Get safe content based on page type

```

```

  let contentHtml = "";
  const tags = pageData.tags || [];
  const views = pageData.views || 0;
  const created = new Date(pageData.created_at).toLocaleDateString();

```

```
switch(pageData.page_type) {
  case 'html':
    // For HTML pages, render the HTML directly (sanitized)
    contentHtml = sanitizeHTML(pageData.content || "");
    break;

  case 'embed':
    // For embed pages
    const embedUrl = pageData.content || "";
    contentHtml = `
      <div class="embed-container">
        <iframe
          src="{${embedUrl}}"
          style="width:100%; height:80vh; border:none; border-radius:15px;"
          allow="camera; microphone; geolocation; fullscreen"
          sandbox="allow-same-origin allow-scripts allow-forms allow-popups allow-modals"
          title="{${pageData.title}}"
        ></iframe>
        <div class="embed-info">
          <strong>Embedded URL:</strong>
          <a href="{${embedUrl}" target="_blank" rel="noopener noreferrer">${embedUrl}</a>
        </div>
      </div>
    `;
    break;

  case 'markdown':
    // For markdown (you would need a markdown parser)
    contentHtml = `<div class="markdown-content">${escapeHTML(pageData.content || "")}</div>`;
    break;

  case 'dashboard':
    // For dashboard with multiple files
    const files = pageData.files || [];
    if (files.length > 0) {
      contentHtml = renderMultiFileDashboard(files, username);
    } else {
      contentHtml = `<div class="single-file">${escapeHTML(pageData.content || "")}</div>`;
    }
    break;

  default:
```

```

    // Plain text or unknown type
    contentHtml = `<div class="plain-content">${escapeHTML(pageData.content || "")}</div>`;
  }

  return `
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>${pageData.title} | @${username} | Pack CDN</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta property="og:title" content="${pageData.title}">
    <meta property="og:description" content="${pageData.description || 'User-created page on
Pack CDN'}">
    <meta property="og:type" content="website">
    <meta property="og:url" content="https://pack-cdn.vercel.app@${username}">
    <style>
      * { margin: 0; padding: 0; box-sizing: border-box; }
      body {
        font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
sans-serif;
        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        min-height: 100vh;
        color: #333;
        line-height: 1.6;
      }
      .container {
        max-width: 1200px;
        margin: 0 auto;
        padding: 20px;
      }
      .page-header {
        background: rgba(255, 255, 255, 0.98);
        border-radius: 20px;
        padding: 40px;
        margin-bottom: 30px;
        box-shadow: 0 15px 50px rgba(0,0,0,0.1);
        text-align: center;
        backdrop-filter: blur(10px);
      }
      .page-content {
        background: white;
        border-radius: 20px;
        padding: 40px;

```

```
    box-shadow: 0 25px 70px rgba(0,0,0,0.1);
    min-height: 500px;
    margin-bottom: 30px;
    overflow-x: auto;
}
.user-badge {
    display: inline-block;
    background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
    color: white;
    padding: 10px 25px;
    border-radius: 30px;
    font-weight: 700;
    font-size: 1.1rem;
    margin: 20px 0;
    text-decoration: none;
    transition: transform 0.3s;
}
.page-stats {
    display: flex;
    gap: 30px;
    justify-content: center;
    margin-top: 25px;
    color: #666;
    font-size: 0.95rem;
    flex-wrap: wrap;
}
.stat-item {
    display: flex;
    align-items: center;
    gap: 8px;
    background: #f8f9fa;
    padding: 10px 20px;
    border-radius: 10px;
}
.action-bar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 30px;
    flex-wrap: wrap;
    gap: 15px;
}
.btn {
    padding: 12px 30px;
```

```
background: white;
border: 2px solid #667eea;
color: #667eea;
border-radius: 12px;
text-decoration: none;
font-weight: 600;
transition: all 0.3s;
display: inline-flex;
align-items: center;
gap: 8px;
}
.btn:hover {
background: #667eea;
color: white;
transform: translateY(-2px);
box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);
}
.btn-create {
background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
color: white;
border: none;
}
}
.tags {
display: flex;
gap: 12px;
flex-wrap: wrap;
margin-top: 25px;
justify-content: center;
}
}
.tag {
background: #e9ecef;
padding: 8px 18px;
border-radius: 20px;
font-size: 0.9rem;
color: #495057;
}
}
.page-footer {
text-align: center;
margin-top: 40px;
padding: 25px;
background: rgba(255, 255, 255, 0.9);
border-radius: 15px;
color: #666;
}
```

```
h1 {
  font-size: 2.8rem;
  margin-bottom: 10px;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
.embed-container {
  margin: 20px 0;
}
.embed-info {
  margin-top: 15px;
  padding: 15px;
  background: #f8f9fa;
  border-radius: 10px;
}

/* Multi-file dashboard styles */
.file-nav {
  background: #f8f9fa;
  border-radius: 10px;
  padding: 20px;
  margin-bottom: 20px;
}
.file-tabs {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
  margin-bottom: 20px;
}
.file-tab {
  padding: 10px 20px;
  background: white;
  border: 2px solid #dee2e6;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s;
}
.file-tab.active {
  background: #667eea;
  color: white;
  border-color: #667eea;
}
.file-content {
```

```

background: white;
border-radius: 10px;
padding: 20px;
min-height: 300px;
}

@media (max-width: 768px) {
  .container { padding: 15px; }
  .page-header, .page-content { padding: 25px; }
  h1 { font-size: 2rem; }
  .action-bar { flex-direction: column; }
}
</style>
</head>
<body>
<div class="container">
  <div class="action-bar">
    <a href="/" class="btn">🏠 Home</a>
    <a href="@random" class="btn btn-create">✨ Create Your Page</a>
    <a href="/api/Pages/Explore/explore-pages" class="btn">🔍 Explore Pages</a>
  </div>

  <div class="page-header">
    <h1>${escapeHTML(pageData.title)}</h1>
    <div class="user-badge">@${username}</div>

    ${pageData.description ? `
      <p style="margin: 20px 0; color: #666; font-size: 1.1rem;">
        ${escapeHTML(pageData.description)}
      </p>
    ` : ""}

    ${tags.length > 0 ? `
      <div class="tags">
        ${tags.map(tag => `<span class="tag">${escapeHTML(tag)}</span>`).join("")}
      </div>
    ` : ""}

    <div class="page-stats">
      <div class="stat-item">👁️ ${views} views</div>
      <div class="stat-item">📅July17 Created ${created}</div>
      <div class="stat-item">🔄 ${pageData.version || 'v1'}</div>
      ${pageData.page_type ? `<div class="stat-item">📄 ${pageData.page_type}</div>` : ""}
    </div>
  </div>
</body>
</html>

```

```

</div>

<div class="page-content" id="content">
  ${contentHtml}
</div>

<div class="page-footer">
  <p> ✨ This page was created with <strong>Pack CDN</strong> •
    <a href="@random">Create your own page</a> •
    <a href="/api/Pages/Explore/explore-pages">Explore more pages</a>
  </p>
  <p style="margin-top: 10px; font-size: 0.9rem; opacity: 0.7;">
    Page ID: ${pageData.page_id} • Last updated: ${new
Date(pageData.updated_at).toLocaleDateString()}
  </p>
</div>
</div>

<script>
  // Execute scripts for HTML pages safely
  if ('${pageData.page_type}' === 'html' || '${pageData.page_type}' === 'dashboard') {
    setTimeout(() => {
      const contentDiv = document.getElementById('content');
      const scripts = contentDiv.getElementsByTagName('script');
      for (let script of scripts) {
        try {
          const newScript = document.createElement('script');
          newScript.textContent = script.textContent;
          document.body.appendChild(newScript);
        } catch (e) {
          console.warn('Script execution skipped:', e.message);
        }
      }
    }, 100);
  }

  // Multi-file tab functionality
  document.addEventListener('click', (e) => {
    if (e.target.classList.contains('file-tab')) {
      const tabs = document.querySelectorAll('.file-tab');
      tabs.forEach(tab => tab.classList.remove('active'));
      e.target.classList.add('active');

      const fileId = e.target.dataset.file;

```

```

        const contents = document.querySelectorAll('.file-content-item');
        contents.forEach(content => {
            content.style.display = content.id === fileId ? 'block' : 'none';
        });
    }
});

    console.log('📄 Page @${username} loaded');
</script>
</body>
</html>
`;
}

// Helper functions
function escapeHTML(text) {
    const div = document.createElement('div');
    div.textContent = text;
    return div.innerHTML;
}

function sanitizeHTML(html) {
    // Basic sanitization - in production use DOMPurify or similar
    return html
        .replace(/<script\b[^\<]*(?:(!<\script><[^\<]*)*<\script>/gi, "")
        .replace(/on\w+="[^\"]*" /g, "")
        .replace(/javascript:/gi, "");
}

function renderMultiFileDashboard(files, username) {
    if (files.length === 0) return '<p>No files in this dashboard.</p>';

    const tabs = files.map((file, index) => `
        <div class="file-tab ${index === 0 ? 'active' : ''}" data-file="file-${index}">
            ${file.name || `File ${index + 1}`} (${file.type || 'html'})
        </div>
    `).join("");

    const contents = files.map((file, index) => `
        <div id="file-${index}" class="file-content-item" style="${index === 0 ? '' : 'display: none;}">
            ${file.type === 'html' ? file.content : `<pre>${escapeHTML(file.content)}</pre>`}
        </div>
    `).join("");

```

```

return `
  <div class="file-nav">
    <h3>📁 Multi-File Dashboard</h3>
    <div class="file-tabs">
      ${tabs}
    </div>
  </div>
  <div class="file-content">
    ${contents}
  </div>
`;
}

```

Embed-Website.js

```

// /api/embed-website.js
export default async function handler(req, res) {
  // Allow CORS for your domain
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');

  if (req.method === 'OPTIONS') {
    return res.status(200).end();
  }

  try {
    const { url, q } = req.query;
    let targetUrl = url;

    // If no URL provided but search query, use DuckDuckGo
    if (!targetUrl && q) {
      targetUrl = `https://duckduckgo.com/html/?q=${encodeURIComponent(q)}`;
    }

    if (!targetUrl) {
      return res.status(400).json({
        error: 'URL parameter required. Use ?url=https://example.com or ?q=search+term'
      });
    }

    // Ensure URL has protocol
    if (!targetUrl.startsWith('http://') && !targetUrl.startsWith('https://')) {
      targetUrl = 'https://' + targetUrl;
    }
  }
}

```

```

// Clean URL for display
const displayUrl = targetUrl.replace(/^https?:\/\//, "");

return res.status(200).json({
  success: true,
  url: targetUrl,
  displayUrl: displayUrl,
  embedUrl: `/embed.html?url=${encodeURIComponent(targetUrl)}`,
  timestamp: new Date().toISOString(),
  type: 'website'
});

} catch (error) {
  console.error('Embed API Error:', error);
  return res.status(500).json({
    error: 'Failed to process embed request',
    message: error.message
  });
}
}

api/Pages/Explore/explore-pages.js
// /api/Pages/Explore/explore-pages.js
import { createClient } from '@supabase/supabase-js'

const supabaseUrl = process.env.SUPABASE_URL
const supabaseKey = process.env.SUPABASE_SERVICE_KEY
const supabase = createClient(supabaseUrl, supabaseKey)

export default async function handler(req, res) {
  // Set CORS headers
  res.setHeader('Access-Control-Allow-Origin', '*')
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS')
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type')

  if (req.method === 'OPTIONS') {
    return res.status(200).end()
  }

  try {
    const {
      page = 1,
      limit = 20,
      sort = 'newest',

```

```
    search = "",
    tag = "",
    type = ""
  } = req.query

  const pageNum = parseInt(page)
  const limitNum = Math.min(parseInt(limit), 100)
  const offset = (pageNum - 1) * limitNum

  // Build query - FIXED: Use 'user_pages' (lowercase) to match your database
  let query = supabase
    .from('user_pages')
    .select('*', { count: 'exact' })
    .eq('is_public', true)

  // Apply filters
  if (search) {
    query =
      query.or(`title.ilike.%${search}%,content.ilike.%${search}%,page_id.ilike.%${search}%`)
  }

  if (tag) {
    query = query.contains('tags', [tag])
  }

  if (type && type !== 'all') {
    query = query.eq('page_type', type)
  }

  // Apply sorting
  switch(sort) {
    case 'popular':
      query = query.order('views', { ascending: false })
      break
    case 'likes':
      query = query.order('likes', { ascending: false })
      break
    case 'oldest':
      query = query.order('created_at', { ascending: true })
      break
    case 'newest':
    default:
      query = query.order('created_at', { ascending: false })
      break
  }
```

```

}

// Add pagination
query = query.range(offset, offset + limitNum - 1)

// Execute query
const { data: pages, error, count } = await query

if (error) throw error

// Get tags for filter
const { data: allTags } = await supabase
  .from('user_pages')
  .select('tags')
  .eq('is_public', true)

const uniqueTags = [...new Set(allTags?.flatMap(p => p.tags || []))].filter(Boolean)

// Get stats
const { count: totalCount } = await supabase
  .from('user_pages')
  .select('*', { count: 'exact', head: true })
  .eq('is_public', true)

const { data: topPages } = await supabase
  .from('user_pages')
  .select('page_id, title, views, created_at')
  .eq('is_public', true)
  .order('views', { ascending: false })
  .limit(5)

// Return HTML page for browser, JSON for API
const accept = req.headers.accept || ''

if (accept.includes('text/html')) {
  return res.send(renderExplorePage({
    pages,
    totalCount,
    pageNum,
    limitNum,
    sort,
    search,
    tag,
    type,
  }

```

```

        uniqueTags,
        topPages
    )))
} else {
    return res.json({
        success: true,
        data: pages,
        meta: {
            page: pageNum,
            limit: limitNum,
            total: count || 0,
            totalPages: Math.ceil((count || 0) / limitNum),
            sort,
            filters: { search, tag, type }
        },
        stats: {
            totalPages: totalCount || 0,
            uniqueTags: uniqueTags.length,
            topPages: topPages || []
        }
    })
}

} catch (error) {
    console.error('Explore pages error:', error)
    return res.status(500).json({
        success: false,
        error: error.message,
        message: 'Failed to fetch pages'
    })
}
}

```

```

function renderExplorePage(data) {
    const {
        pages = [],
        totalCount = 0,
        pageNum = 1,
        limitNum = 20,
        sort = 'newest',
        search = "",
        tag = "",
        type = "",
        uniqueTags = [],
    } = data || {}
    return {
        data: {
            pages,
            meta: {
                page: pageNum,
                limit: limitNum,
                total: totalCount,
                totalPages: Math.ceil(totalCount / limitNum),
                sort,
                filters: { search, tag, type }
            },
            stats: {
                totalPages: totalCount,
                uniqueTags: uniqueTags.length,
                topPages: topPages || []
            }
        },
        error: null,
        message: null
    }
}

```

```

    topPages = []
  } = data

const totalPages = Math.ceil(totalCount / limitNum)

// Helper function to escape HTML
function escapeHtml(text) {
  if (!text) return "";
  return text
    .replace(/&/g, "&amp;")
    .replace(/</g, "&lt;")
    .replace(/>/g, "&gt;")
    .replace(/"/g, "&quot;")
    .replace(/'/g, "&#039;");
}

// Helper function to truncate text
function truncate(text, length) {
  if (!text) return "";
  if (text.length <= length) return text;
  return text.substring(0, length) + '...';
}

return `
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Explore User Pages | Pack CDN</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      min-height: 100vh;
      color: #333;
    }

```

```
.container {
  max-width: 1400px;
  margin: 0 auto;
  padding: 20px;
}

.header {
  text-align: center;
  margin-bottom: 40px;
  color: white;
}

.header h1 {
  font-size: 3rem;
  margin-bottom: 10px;
  text-shadow: 0 2px 10px rgba(0,0,0,0.2);
}

.header p {
  font-size: 1.2rem;
  opacity: 0.9;
}

.main-content {
  display: grid;
  grid-template-columns: 300px 1fr;
  gap: 30px;
}

@media (max-width: 1024px) {
  .main-content {
    grid-template-columns: 1fr;
  }
}

/* Filters Sidebar */
.sidebar {
  background: rgba(255, 255, 255, 0.95);
  border-radius: 20px;
  padding: 30px;
  box-shadow: 0 15px 40px rgba(0,0,0,0.1);
  backdrop-filter: blur(10px);
  height: fit-content;
  position: sticky;
```

```
    top: 20px;
}

.filter-group {
  margin-bottom: 30px;
}

.filter-title {
  font-size: 1.1rem;
  font-weight: 600;
  margin-bottom: 15px;
  color: #444;
  display: flex;
  align-items: center;
  gap: 8px;
}

.search-box {
  width: 100%;
  padding: 15px;
  border: 2px solid #e0e0e0;
  border-radius: 12px;
  font-size: 1rem;
  transition: all 0.3s;
  margin-bottom: 20px;
}

.search-box:focus {
  outline: none;
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.sort-options {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.sort-option {
  padding: 12px 20px;
  background: white;
  border: 2px solid #e0e0e0;
  border-radius: 10px;
```

```
    cursor: pointer;
    transition: all 0.3s;
    text-align: left;
    font-size: 0.95rem;
}

.sort-option:hover {
    border-color: #667eea;
    background: #f8f9ff;
}

.sort-option.active {
    background: #667eea;
    color: white;
    border-color: #667eea;
}

.tag-cloud {
    display: flex;
    flex-wrap: wrap;
    gap: 8px;
}

.tag {
    padding: 8px 16px;
    background: #f0f2ff;
    border: 1px solid #d0d7ff;
    border-radius: 20px;
    font-size: 0.85rem;
    cursor: pointer;
    transition: all 0.3s;
}

.tag:hover, .tag.active {
    background: #667eea;
    color: white;
    border-color: #667eea;
}

.type-filter {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
}
```

```
.type-btn {
  padding: 10px;
  background: white;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  cursor: pointer;
  text-align: center;
  font-size: 0.9rem;
  transition: all 0.3s;
}

.type-btn:hover, .type-btn.active {
  background: #667eea;
  color: white;
  border-color: #667eea;
}

/* Pages Grid */
.pages-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(350px, 1fr));
  gap: 25px;
}

.page-card {
  background: white;
  border-radius: 20px;
  overflow: hidden;
  box-shadow: 0 10px 30px rgba(0,0,0,0.1);
  transition: all 0.3s;
  height: 100%;
  display: flex;
  flex-direction: column;
}

.page-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 20px 40px rgba(0,0,0,0.15);
}

.card-header {
  padding: 25px;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
```

```
    color: white;
}

.page-title {
  font-size: 1.4rem;
  margin-bottom: 10px;
  line-height: 1.3;
}

.page-id {
  font-size: 0.9rem;
  opacity: 0.9;
  font-family: monospace;
}

.card-body {
  padding: 25px;
  flex-grow: 1;
  display: flex;
  flex-direction: column;
}

.page-preview {
  background: #f8f9fa;
  border-radius: 10px;
  padding: 20px;
  margin-bottom: 20px;
  flex-grow: 1;
  overflow: hidden;
}

.page-preview-content {
  max-height: 150px;
  overflow: hidden;
  line-height: 1.5;
  color: #555;
}

.page-stats {
  display: flex;
  gap: 15px;
  margin-bottom: 20px;
  font-size: 0.9rem;
  color: #666;
```

```
}

.stat {
  display: flex;
  align-items: center;
  gap: 5px;
}

.card-footer {
  padding: 0 25px 25px;
  display: flex;
  gap: 10px;
}

.btn {
  padding: 10px 20px;
  border-radius: 10px;
  text-decoration: none;
  font-weight: 600;
  transition: all 0.3s;
  flex: 1;
  text-align: center;
}

.btn-primary {
  background: #667eea;
  color: white;
}

.btn-primary:hover {
  background: #5a67d8;
}

.btn-secondary {
  background: #f8f9fa;
  color: #333;
  border: 2px solid #e0e0e0;
}

.btn-secondary:hover {
  background: #e9ecef;
}

/* Pagination */
```

```
.pagination {
  display: flex;
  justify-content: center;
  gap: 10px;
  margin-top: 40px;
  flex-wrap: wrap;
}

.page-btn {
  padding: 10px 18px;
  background: white;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s;
}

.page-btn:hover {
  border-color: #667eea;
  color: #667eea;
}

.page-btn.active {
  background: #667eea;
  color: white;
  border-color: #667eea;
}

/* Stats */
.stats-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 20px;
  margin-bottom: 40px;
}

.stat-card {
  background: rgba(255, 255, 255, 0.9);
  border-radius: 15px;
  padding: 25px;
  text-align: center;
  backdrop-filter: blur(10px);
}
```

```
.stat-number {
  font-size: 2.5rem;
  font-weight: 700;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  margin-bottom: 10px;
}
```

```
.stat-label {
  color: #666;
  font-size: 0.95rem;
}
```

/* Empty State */

```
.empty-state {
  text-align: center;
  padding: 80px 20px;
  background: white;
  border-radius: 20px;
  grid-column: 1 / -1;
}
```

```
.empty-icon {
  font-size: 4rem;
  margin-bottom: 20px;
  opacity: 0.3;
}
```

/* Top Pages */

```
.top-pages {
  margin-top: 40px;
  padding: 25px;
  background: rgba(255, 255, 255, 0.9);
  border-radius: 20px;
}
```

```
.top-pages-title {
  font-size: 1.3rem;
  margin-bottom: 20px;
  color: #444;
}
```

```
.top-page-item {
```

```
display: flex;
justify-content: space-between;
align-items: center;
padding: 15px;
background: white;
border-radius: 10px;
margin-bottom: 10px;
transition: all 0.3s;
}

.top-page-item:hover {
  transform: translateX(5px);
  box-shadow: 0 5px 15px rgba(0,0,0,0.1);
}

.action-bar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 30px;
  flex-wrap: wrap;
  gap: 15px;
}

.create-btn {
  padding: 15px 30px;
  background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
  color: white;
  border: none;
  border-radius: 12px;
  text-decoration: none;
  font-weight: 600;
  display: inline-flex;
  align-items: center;
  gap: 8px;
  transition: all 0.3s;
}

.create-btn:hover {
  transform: translateY(-2px);
  box-shadow: 0 10px 20px rgba(245, 87, 108, 0.3);
}

@media (max-width: 768px) {
```

```

    .pages-grid {
      grid-template-columns: 1fr;
    }

    .header h1 {
      font-size: 2.2rem;
    }

    .type-filter {
      grid-template-columns: 1fr;
    }
  }
</style>
</head>
<body>
  <div class="container">
    <div class="header">
      <h1>✨ Explore User Pages</h1>
      <p>Discover amazing pages created by the Pack CDN community</p>
    </div>

    <div class="action-bar">
      <div>
        <a href="/" class="btn btn-secondary">🏠 Home</a>
        <a href="@random" class="create-btn">✨ Create Your Page</a>
      </div>
      <div class="stat-card" style="padding: 15px 25px;">
        <div style="font-size: 1.5rem; font-weight: 700; color: #667eea;">${totalCount}</div>
        <div style="color: #666;">Total Pages</div>
      </div>
    </div>

    <div class="stats-grid">
      <div class="stat-card">
        <div class="stat-number">${pages.length}</div>
        <div class="stat-label">Displayed Pages</div>
      </div>
      <div class="stat-card">
        <div class="stat-number">${uniqueTags.length}</div>
        <div class="stat-label">Unique Tags</div>
      </div>
      <div class="stat-card">
        <div class="stat-number">${Math.ceil(totalCount / limitNum)}</div>
        <div class="stat-label">Total Pages</div>
      </div>
    </div>
  </div>
</body>
</html>

```

```

    </div>
    <div class="stat-card">
      <div class="stat-number">${topPages.reduce((sum, p) => sum + (p.views || 0),
0).toLocaleString())}</div>
      <div class="stat-label">Total Views</div>
    </div>
  </div>

  <div class="main-content">
    <!-- Sidebar Filters -->
    <div class="sidebar">
      <input
        type="text"
        class="search-box"
        placeholder="Search pages..."
        value="${escapeHtml(search)}"
        onkeyup="if(event.key==='Enter') searchPages()"
        id="searchInput"
      >

      <div class="filter-group">
        <div class="filter-title"><img alt="Bar chart icon" data-bbox="398 478 421 493"/> Sort By</div>
        <div class="sort-options">
          <button class="sort-option ${sort === 'newest' ? 'active' : ''}"
onclick="setSort('newest')">
            <img alt="Newest icon" data-bbox="258 553 278 568"/> Newest First
          </button>
          <button class="sort-option ${sort === 'oldest' ? 'active' : ''}"
onclick="setSort('oldest')">
            <img alt="Calendar icon" data-bbox="258 623 278 638"/> Oldest First
          </button>
          <button class="sort-option ${sort === 'popular' ? 'active' : ''}"
onclick="setSort('popular')">
            <img alt="Fire icon" data-bbox="258 698 278 713"/> Most Popular
          </button>
          <button class="sort-option ${sort === 'likes' ? 'active' : ''}"
onclick="setSort('likes')">
            <img alt="Heart icon" data-bbox="258 773 278 788"/> Most Liked
          </button>
        </div>
      </div>

      <div class="filter-group">
        <div class="filter-title"><img alt="Tag icon" data-bbox="398 883 421 898"/> Filter by Tag</div>

```

```

<div class="tag-cloud">
  <button class="tag ${!tag ? 'active' : ''}" onclick="setTag('')">
    All Tags
  </button>
  ${uniqueTags.slice(0, 20).map(t => `
    <button class="tag ${tag === t ? 'active' : ''}"
onclick="setTag('${escapeHtml(t)}')">
      ${escapeHtml(t)}
    </button>
  `).join("")}
  ${uniqueTags.length > 20 ? `
    <button class="tag" onclick="showAllTags()">
      +${uniqueTags.length - 20} more...
    </button>
  ` : ''}
</div>
</div>

<div class="filter-group">
  <div class="filter-title"> 📄 Page Type</div>
  <div class="type-filter">
    <button class="type-btn ${!type ? 'active' : ''}" onclick="setType('')">
      All Types
    </button>
    <button class="type-btn ${type === 'html' ? 'active' : ''}" onclick="setType('html')">
      HTML
    </button>
    <button class="type-btn ${type === 'embed' ? 'active' : ''}"
onclick="setType('embed')">
      Embed
    </button>
    <button class="type-btn ${type === 'markdown' ? 'active' : ''}"
onclick="setType('markdown')">
      Markdown
    </button>
    <button class="type-btn ${type === 'dashboard' ? 'active' : ''}"
onclick="setType('dashboard')">
      Dashboard
    </button>
  </div>
</div>

  <button class="btn btn-primary" style="width: 100%; margin-top: 20px;"
onclick="applyFilters()">

```

```

    <button class="btn btn-secondary" style="width: 100%; margin-top: 10px;"
onclick="resetFilters()">
      <img alt="Reset icon" data-bbox="146 118 176 138"/> Reset
    </button>
  </div>

<!-- Pages Grid -->
<div>
  ${pages.length === 0 ? `
    <div class="empty-state">
      <div class="empty-icon"><img alt="Empty state icon" data-bbox="451 300 476 320"/></div>
      <h2 style="margin-bottom: 10px; color: #666;">No pages found</h2>
      <p style="color: #888; margin-bottom: 30px;">
        ${search || tag || type ? 'Try changing your filters' : 'Be the first to create a
page!'}
      </p>
      <a href="@random" class="create-btn">✨ Create First Page</a>
    </div>
  ` : `
    <div class="pages-grid">
      ${pages.map(page => {
        const pageTitle = page.title || 'Untitled Page';
        const pageId = page.page_id || '';
        const pageContent = page.content || '';
        const pageType = page.page_type || 'html';
        const pageTags = page.tags || [];
        const pageViews = page.views || 0;
        const pageLikes = page.likes || 0;
        const createdAt = page.created_at ? new
Date(page.created_at).toLocaleDateString() : 'Unknown';

        return `
          <div class="page-card">
            <div class="card-header">
              <div class="page-title">${escapeHtml(pageTitle)}</div>
              <div class="page-id">@${escapeHtml(pageId)}</div>
            </div>
            <div class="card-body">
              <div class="page-preview">
                <div class="page-preview-content">
                  ${pageType === 'embed'

```

```

        ? `🌐 Embedded: ${truncate(escapeHtml(pageContent), 100)}`
        : truncate(escapeHtml(pageContent), 200)}
    </div>
</div>
<div class="page-stats">
    <div class="stat">👁️ ${pageViews}</div>
    <div class="stat">❤️ ${pageLikes}</div>
    <div class="stat">📅 day17 ${createdDate}</div>
    <div class="stat">🏷️ ${escapeHtml(pageType)}</div>
</div>
${pageTags.length > 0 ? `
    <div style="display: flex; gap: 8px; flex-wrap: wrap; margin-bottom:
15px;">
        ${pageTags.slice(0, 3).map(t => `
            <span style="background: #f0f2ff; padding: 4px 10px;
border-radius: 12px; font-size: 0.8rem; color: #667eea;">
                ${escapeHtml(t)}
            </span>
        `).join("")}
        ${pageTags.length > 3 ? `
            <span style="background: #f0f2ff; padding: 4px 10px;
border-radius: 12px; font-size: 0.8rem; color: #888;">
                +${pageTags.length - 3}
            </span>
        ` : ""}
    </div>
    ` : ""}
</div>
<div class="card-footer">
    <a href="@${escapeHtml(pageId)}" class="btn btn-primary">Visit
Page</a>
    <button class="btn btn-secondary"
onclick="likePage('${escapeHtml(pageId)}')">❤️ Like</button>
</div>
</div>
`;
}).join("")}
</div>

${totalPages > 1 ? `
    <div class="pagination">
        ${pageNum > 1 ? `
            <button class="page-btn" onclick="goToPage(${pageNum - 1})">←
Previous</button>

```

```

    ` : "}"

    ${Array.from({ length: Math.min(5, totalPages) }, (_, i) => {
      let pageToShow = i + 1;
      if (totalPages > 5) {
        if (pageNum <= 3) pageToShow = i + 1;
        else if (pageNum >= totalPages - 2) pageToShow = totalPages - 4 + i;
        else pageToShow = pageNum - 2 + i;
      }
      return pageToShow <= totalPages ? `
        <button class="page-btn ${pageNum === pageToShow ? 'active' : ""}
          onclick="goToPage(${pageToShow})">
            ${pageToShow}
          </button>
        ` : "";
    }).join("")}

    ${pageNum < totalPages ? `
      <button class="page-btn" onclick="goToPage(${pageNum + 1})">Next
    →</button>
    ` : "}"
  </div>
  ` : "}"
}

${topPages.length > 0 ? `
  <div class="top-pages">
    <div class="top-pages-title">🔥 Trending Pages</div>
    ${topPages.map(page => {
      const pageTitle = page.title || 'Untitled';
      const pageId = page.page_id || "";
      const pageViews = page.views || 0;
      return `
        <div class="top-page-item">
          <div>
            <div style="font-weight: 600;">${escapeHtml(pageTitle)}</div>
            <div style="font-size: 0.9rem; color: #666;">@${escapeHtml(pageId)} •
${pageViews} views</div>
          </div>
          <div>
            <a href="@${escapeHtml(pageId)}" class="btn btn-primary"
style="padding: 8px 16px; font-size: 0.9rem;">
              Visit
            </a>
          </div>
        </div>
      `
    })}
  </div>
`
}

```

```

        </div>
    </div>
    `;
    }).join("")
</div>
    ` : ""
</div>
</div>
</div>

<script>
    let currentFilters = {
        page: ${pageNum},
        limit: ${limitNum},
        sort: '${escapeHtml(sort)}',
        search: '${escapeHtml(search)}',
        tag: '${escapeHtml(tag)}',
        type: '${escapeHtml(type)}'
    }

    function updateURL() {
        const params = new URLSearchParams()
        Object.entries(currentFilters).forEach(([key, value]) => {
            if (value) params.set(key, value)
        })
        const url = '/api/Pages/Explore/explore-pages' + (params.toString() ? '?' +
params.toString() : "")
        window.history.pushState({}, "", url)
    }

    function applyFilters() {
        currentFilters.search = document.getElementById('searchInput').value
        currentFilters.page = 1
        updateURL()
        window.location.reload()
    }

    function setSort(sortBy) {
        currentFilters.sort = sortBy
        currentFilters.page = 1
        updateURL()
        window.location.reload()
    }

```

```
function setTag(selectedTag) {
  currentFilters.tag = selectedTag
  currentFilters.page = 1
  updateURL()
  window.location.reload()
}
```

```
function setType(selectedType) {
  currentFilters.type = selectedType
  currentFilters.page = 1
  updateURL()
  window.location.reload()
}
```

```
function resetFilters() {
  currentFilters = {
    page: 1,
    limit: ${limitNum},
    sort: 'newest',
    search: "",
    tag: "",
    type: ""
  }
  updateURL()
  window.location.reload()
}
```

```
function goToPage(page) {
  currentFilters.page = page
  updateURL()
  window.location.reload()
}
```

```
function searchPages() {
  applyFilters()
}
```

```
function showAllTags() {
  alert('All tags: ${uniqueTags.map(t => escapeHtml(t)).join(', ')}')
}
```

```
async function likePage(pageId) {
  try {
    const response = await fetch('/api/create-page', {
```

```

        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-Action': 'like'
        },
        body: JSON.stringify({ page_id: pageId })
    })

    const data = await response.json();

    if (response.ok) {
        alert('Page liked! ❤️')
        // Refresh to show updated likes
        window.location.reload()
    } else {
        alert(data.error || 'Failed to like page')
    }
} catch (error) {
    alert('Failed to like page: ' + error.message)
}
}

// Handle Enter key in search
document.getElementById('searchInput').addEventListener('keypress', (e) => {
    if (e.key === 'Enter') searchPages()
})

// Auto-focus search on page load
window.addEventListener('load', () => {
    if (currentFilters.search) {
        document.getElementById('searchInput').focus()
        document.getElementById('searchInput').select()
    }
})

// Keyboard shortcuts
document.addEventListener('keydown', (e) => {
    if (e.ctrlKey && e.key === 'f') {
        e.preventDefault()
        document.getElementById('searchInput').focus()
        document.getElementById('searchInput').select()
    }
    if (e.key === 'Escape' && document.getElementById('searchInput') ===
document.activeElement) {

```

```

        document.getElementById('searchInput').value = ""
    }
    })
</script>
</body>
</html>
`
}

```

[Admin-Auth.js](#)

```

// Use ESM syntax for Vercel
import { parse } from 'cookie';
import fs from 'fs/promises';
import path from 'path';
import { fileURLToPath } from 'url';

// Get __dirname equivalent in ES modules
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

export default async function handler(req, res) {
    const correctPassword = process.env.PASSWORD;

    // Check if password is provided via query parameter
    const providedPassword = req.query.password;

    // Parse cookies from headers
    const cookies = req.headers.cookie ? parse(req.headers.cookie) : {};
    const isAuthenticated = cookies.admin_auth === 'true';

    // If authenticated via cookie, serve the admin page
    if (isAuthenticated) {
        try {
            const adminHtmlPath = path.join(__dirname, '..', 'Private', 'admin.html');
            const adminHtml = await fs.readFile(adminHtmlPath, 'utf8');

            res.setHeader('Content-Type', 'text/html');
            return res.status(200).send(adminHtml);
        } catch (error) {
            console.error('Error reading admin.html:', error);
            return res.status(404).send('Admin page not found');
        }
    }
}

```

```

// Check if password is being submitted
if (providedPassword) {
  if (providedPassword === correctPassword) {
    // Set authentication cookie (expires in 1 hour, Secure flag for HTTPS, SameSite=Lax)
    res.setHeader('Set-Cookie', [
      'admin_auth=true; Path=/; HttpOnly; Max-Age=3600; SameSite=Lax'
    ].join('; '));

    // Remove password from URL and redirect
    const url = new URL(req.url, `https://${req.headers.host}`);
    url.searchParams.delete('password');
    const redirectUrl = url.pathname + url.search;

    res.writeHead(302, { 'Location': redirectUrl });
    return res.end();
  } else {
    // Show error message
    return showPasswordForm(res, 'Incorrect password. Please try again.');
```

```

  }
}
```

```

// Show password form
return showPasswordForm(res);
}
```

```

function showPasswordForm(res, errorMessage = "") {
  const html = `
    <!DOCTYPE html>
    <html>
    <head>
      <title>Admin Access Required</title>
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <style>
        body {
          font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
sans-serif;
          background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
          display: flex;
          justify-content: center;
          align-items: center;
          height: 100vh;
          margin: 0;
        }
      </style>
    </head>
    <body>
      <div>
        <h1>Admin Access Required</h1>
        <p>${errorMessage}</p>
      </div>
    </body>
  `;
  res.send(html);
}
```

```
.login-container {
  background: rgba(255, 255, 255, 0.95);
  padding: 40px;
  border-radius: 20px;
  box-shadow: 0 20px 60px rgba(0, 0, 0, 0.3);
  width: 90%;
  max-width: 400px;
  text-align: center;
  backdrop-filter: blur(10px);
}

.logo {
  font-size: 2.5em;
  margin-bottom: 20px;
  color: #333;
}

h1 {
  color: #333;
  margin-bottom: 30px;
  font-weight: 300;
}

.password-input {
  width: 100%;
  padding: 15px;
  border: 2px solid #ddd;
  border-radius: 10px;
  font-size: 16px;
  margin-bottom: 20px;
  box-sizing: border-box;
  transition: border-color 0.3s;
}

.password-input:focus {
  outline: none;
  border-color: #667eea;
}

.submit-btn {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  border: none;
  padding: 15px 30px;
```

```

border-radius: 10px;
font-size: 16px;
cursor: pointer;
width: 100%;
transition: transform 0.2s, box-shadow 0.2s;
}

.submit-btn:hover {
transform: translateY(-2px);
box-shadow: 0 10px 20px rgba(102, 126, 234, 0.4);
}

.error-message {
color: #e74c3c;
background: rgba(231, 76, 60, 0.1);
padding: 10px;
border-radius: 5px;
margin-bottom: 20px;
}

.hint {
margin-top: 20px;
color: #666;
font-size: 14px;
}
</style>
</head>
<body>
<div class="login-container">
<div class="logo">🔒 </div>
<h1>Admin Access Required</h1>

${errorMessage ? `<div class="error-message">${errorMessage}</div>` : ""}

<form method="GET" action="">
<input
type="password"
name="password"
class="password-input"
placeholder="Enter admin password"
required
autocomplete="current-password"
autofocus
>

```

```
    <button type="submit" class="submit-btn">Access Admin Panel</button>
  </form>
```

```
    <div class="hint">
      Contact system administrator for credentials
    </div>
  </div>
</body>
</html>
```

```
`;
```

```
res.setHeader('Content-Type', 'text/html');
res.status(401).send(html);
}
```

[Analyze.js](#)

```
// /api/analyze.js
export default async function handler(req, res) {
  if (req.method !== 'POST') {
    return res.status(405).json({ error: 'Method not allowed' });
  }

  const { code, packageName, packageType } = req.body;

  try {
    // Generate pack.json based on code analysis
    const packJson = {
      name: packageName,
      version: "1.0.0",
      type: packageType,
      dependencies: analyzeDependencies(code, packageType),
      entryPoints: findEntryPoints(code, packageType),
      files: [],
      createdAt: new Date().toISOString()
    };

    res.status(200).json({
      success: true,
      packJson,
      fileStructure: suggestFileStructure(code, packageType)
    });
  } catch (error) {
    res.status(500).json({ error: 'Analysis failed' });
  }
}
```

```

}

function analyzeDependencies(code, type) {
  const deps = {};

  if (type === 'npm') {
    // Find imports in JS/TS
    const importRegex = /from\s+["']([^\"]+)"'|require\(["']([^\"]+)"'\)/g;
    const matches = [...code.matchAll(importRegex)];
    matches.forEach(match => {
      const dep = match[1] || match[2];
      if (!dep.startsWith('.')) {
        const depName = dep.split('/')[0];
        deps[depName] = 'latest';
      }
    });
  } else if (type === 'python') {
    // Find imports in Python
    const importRegex = /^(?:import|from)\s+([\w\.\s]+)/gm;
    const matches = [...code.matchAll(importRegex)];
    matches.forEach(match => deps[match[1]] = '*');
  }

  return deps;
}

function findEntryPoints(code, type) {
  if (type === 'npm') {
    return code.includes('export default') ? ['index.js'] : ['main.js'];
  } else if (type === 'python') {
    return code.includes('def main') ? ['main.py'] : ['app.py'];
  }
  return ['index.js'];
}

function suggestFileStructure(code, type) {
  const structure = [];
  const fileName = type === 'python' ? 'main.py' : 'index.js';

  structure.push({
    path: fileName,
    content: code,
    isEntry: true
  });
}

```

```

if (type === 'npm' && code.includes('export')) {
  structure.push({
    path: 'package.json',
    content: JSON.stringify({
      name: 'temp-package',
      version: '1.0.0',
      main: 'index.js'
    }, null, 2),
    isEntry: false
  });
}

return structure;
}

```

[Get-Pack.js](#)

```

import { createClient } from '@supabase/supabase-js';

const supabase = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_ANON_KEY
);

export default async function handler(req, res) {
  const { id } = req.query;

  if (!id) {
    return res.status(400).json({ error: 'Missing pack ID' });
  }

  try {
    console.log(`Looking for pack with ID: ${id}`);

    // First try to find by url_id (short ID)
    let { data, error } = await supabase
      .from('packs')
      .select('*')
      .eq('url_id', id)
      .single();

    // If not found by url_id, try by id (UUID)
    if (error && error.code === 'PGRST116') {
      console.log(`Not found by url_id, trying by UUID id: ${id}`);

```

```

const { data: uuidData, error: uuidError } = await supabase
  .from('packs')
  .select('*')
  .eq('id', id)
  .single();

if (uuidError) {
  if (uuidError.code === 'PGRST116') {
    return res.status(404).json({
      success: false,
      error: 'Pack not found',
      debug: `No pack found with ID: ${id}`
    });
  }
  throw uuidError;
}

data = uuidData;
console.log(`Found pack by UUID: ${data.name || data.id}`);
} else if (error) {
  throw error;
} else {
  console.log(`Found pack by url_id: ${data.name || data.id}`);
}

if (!data) {
  return res.status(404).json({
    success: false,
    error: 'Pack not found',
    debug: `No data returned for ID: ${id}`
  });
}

console.log(`Pack details - Name: ${data.name}, url_id: ${data.url_id}, id: ${data.id}`);

// Parse pack_json if it's a string
let packJson = data.pack_json;
if (typeof packJson === 'string') {
  try {
    packJson = JSON.parse(packJson);
  } catch (e) {
    console.warn('Failed to parse pack_json:', e);
  }
}

```

```
// Return the pack data
res.status(200).json({
  success: true,
  pack: {
    ...data,
    pack_json: packJson
  }
});
} catch (error) {
  console.error('Get pack error:', error);
  res.status(500).json({
    success: false,
    error: 'Failed to get pack: ' + error.message,
    debug: 'Check server logs for details'
  });
}
}
```

[Search.js](#)

```
// /api/search.js
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_ANON_KEY
);

// Search scoring weights
const SEARCH_WEIGHTS = {
  NAME_MATCH: 10,
  DESCRIPTION_MATCH: 5,
  KEYWORD_MATCH: 3,
  POPULARITY: 2,
  RECENCY: 1
};

// Available filters
const AVAILABLE_FILTERS = {
  type: ['basic', 'standard', 'advanced'],
  language: ['javascript', 'python', 'wasm', 'json', 'markdown', 'mixed'],
  minVersion: null,
  maxVersion: null,
  hasReadme: null,
}
```

```
hasLicense: null,
hasTests: null,
verified: null,
dependency: null,
author: null,
minDownloads: null,
minViews: null,
sort: ['relevance', 'popular', 'newest', 'updated', 'name']
};

export default async function handler(req, res) {
  // CORS headers
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'GET, OPTIONS');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');

  if (req.method === 'OPTIONS') {
    return res.status(200).end();
  }

  if (req.method !== 'GET') {
    return res.status(405).json({
      success: false,
      error: 'Method not allowed'
    });
  }

  try {
    const {
      q,
      type,
      language,
      minVersion,
      maxVersion,
      hasReadme,
      hasLicense,
      hasTests,
      verified,
      dependency,
      author,
      minDownloads = 0,
      minViews = 0,
      sort = 'relevance',
      page = 1,

```

```

    limit = 20,
    includeMetadata = 'true',
    advanced = 'false'
  } = req.query;

const pageNum = parseInt(page);
const limitNum = parseInt(limit) > 100 ? 100 : parseInt(limit);
const offset = (pageNum - 1) * limitNum;
const useAdvancedSearch = advanced === 'true';

// Build base query - start with packs table for backward compatibility
let query = supabase
  .from('packs')
  .select(`
    id,
    url_id,
    name,
    pack_json,
    version,
    cdn_url,
    worker_url,
    is_public,
    created_at,
    updated_at,
    views,
    downloads,
    publisher_id,
    package_type,
    last_accessed,
    publish_ip,
    encrypted_key,
    files
  `, { count: 'exact' })
  .eq('is_public', true);

// Apply basic filters (backward compatible)
if (q) {
  // Enhanced search: search in name, pack_json (description), and keywords
  const searchTerms = q.toLowerCase().trim();

  // We'll use multiple OR conditions for better search
  query = query.or(`
    name.ilike.%${searchTerms}%,
    pack_json->>'description'.ilike.%${searchTerms}%,

```

```

    pack_json->>'keywords'.ilike.%%${searchTerms}%,
    pack_json->>'author'.ilike.%%${searchTerms}%
`);
}

// Package type filter
if (type && ['basic', 'standard', 'advanced'].includes(type)) {
    query = query.eq('package_type', type);
}

// Version range filter
if (minVersion) {
    query = query.gte('version', minVersion);
}
if (maxVersion) {
    query = query.lte('version', maxVersion);
}

// Boolean filters
if (hasReadme === 'true') {
    query = query.ilike('pack_json', '%README%');
}

if (hasLicense === 'true') {
    query = query.ilike('pack_json', '%LICENSE%');
}

if (hasTests === 'true') {
    query = query.or('name.ilike.%test%,name.ilike.%spec%');
}

// Author/Publisher filter
if (author) {
    query = query.or(`
        pack_json->>'author'.ilike.%%${author}%,
        publisher_id.ilike.%%${author}%
    `);
}

// Popularity filters
if (parseInt(minDownloads) > 0) {
    query = query.gte('downloads', parseInt(minDownloads));
}

```

```
if (parseInt(minViews) > 0) {
  query = query.gte('views', parseInt(minViews));
}

// Sorting
switch (sort) {
  case 'popular':
    query = query.order('downloads', { ascending: false });
    break;
  case 'newest':
    query = query.order('created_at', { ascending: false });
    break;
  case 'updated':
    query = query.order('updated_at', { ascending: false });
    break;
  case 'name':
    query = query.order('name', { ascending: true });
    break;
  case 'relevance':
  default:
    // Default: sort by relevance (downloads + recency)
    if (q) {
      // When searching, we want relevance-based sorting
      // We'll handle this after fetching results
      query = query.order('downloads', { ascending: false });
    } else {
      query = query.order('created_at', { ascending: false });
    }
    break;
}

// Get total count first for pagination
const { count, error: countError } = await query;

if (countError) throw countError;

// Apply pagination
query = query.range(offset, offset + limitNum - 1);

// Execute query for packs
const { data: packs, error } = await query;

if (error) throw error;
```

```

// If no packs found, return empty
if (!packs || packs.length === 0) {
  return res.status(200).json({
    success: true,
    packs: [],
    pagination: {
      page: pageNum,
      limit: limitNum,
      total: 0,
      totalPages: 0,
      hasNextPage: false,
      hasPreviousPage: pageNum > 1
    },
    filters: {
      applied: {
        q: q || null,
        type: type || null,
        language: language || null,
        sort: sort || 'relevance'
      },
      available: AVAILABLE_FILTERS
    },
    metadata: {
      processingTime: Date.now(),
      advancedSearch: useAdvancedSearch,
      includeMetadata: includeMetadata === 'true'
    }
  });
}

```

```

// Get pack IDs for fetching additional data
const packIds = packs.map(pack => pack.id);

```

```

// Fetch additional data from other tables if advanced search is enabled
let metadataMap = {};
let versionsMap = {};
let collaboratorsMap = {};
let dependenciesMap = {};
let changesMap = {};

```

```

if (useAdvancedSearch) {
  // Fetch metadata
  try {
    const { data: metadata, error: metaError } = await supabase

```

```

    .from('pack_metadata')
    .select('*')
    .in('pack_id', packIds);

    if (!metaError && metadata) {
      metadata.forEach(meta => {
        metadataMap[meta.pack_id] = meta;
      });
    }
  } catch (metaErr) {
    console.warn('Error fetching metadata:', metaErr.message);
  }

  // Fetch versions
  try {
    const { data: versions, error: versionsError } = await supabase
      .from('pack_versions')
      .select('*')
      .in('pack_id', packIds)
      .order('version_number', { ascending: false });

    if (!versionsError && versions) {
      versionsMap = versions.reduce((acc, version) => {
        if (!acc[version.pack_id]) {
          acc[version.pack_id] = [];
        }
        acc[version.pack_id].push(version);
        return acc;
      }, {});
    }
  } catch (versionsErr) {
    console.warn('Error fetching versions:', versionsErr.message);
  }

  // Fetch collaborators
  try {
    const { data: collaborators, error: collabError } = await supabase
      .from('pack_collaborators')
      .select('*')
      .in('pack_id', packIds);

    if (!collabError && collaborators) {
      collaboratorsMap = collaborators.reduce((acc, collab) => {
        if (!acc[collab.pack_id]) {

```

```

        acc[collab.pack_id] = [];
      }
      acc[collab.pack_id].push(collab);
      return acc;
    }, {});
  }
} catch (collabErr) {
  console.warn('Error fetching collaborators:', collabErr.message);
}

// Fetch dependencies if dependency filter is active
if (dependency) {
  try {
    const { data: deps, error: depError } = await supabase
      .from('pack_dependencies')
      .select('*')
      .in('pack_id', packIds)
      .ilike('dependency_name', `%${dependency}%`);

    if (!depError && deps) {
      deps.forEach(dep => {
        if (!dependenciesMap[dep.pack_id]) {
          dependenciesMap[dep.pack_id] = [];
        }
        dependenciesMap[dep.pack_id].push(dep.dependency_name);
      });
    }
  } catch (depErr) {
    console.warn('Error fetching dependencies:', depErr.message);
  }
}

// Fetch recent changes
try {
  const { data: changes, error: changesError } = await supabase
    .from('pack_changes')
    .select('*')
    .in('pack_id', packIds)
    .order('created_at', { ascending: false })
    .limit(3 * packIds.length); // Get recent changes for all packs

  if (!changesError && changes) {
    changesMap = changes.reduce((acc, change) => {
      if (!acc[change.pack_id]) {

```

```

        acc[change.pack_id] = [];
    }
    if (acc[change.pack_id].length < 3) { // Limit to 3 recent changes per pack
        acc[change.pack_id].push(change);
    }
    return acc;
}, {});
}
} catch (changesErr) {
    console.warn('Error fetching changes:', changesErr.message);
}
}
}

```

```

// Process and enhance results
const enhancedPacks = packs.map(pack => {
    const enhancedPack = {
        id: pack.id,
        urlId: pack.url_id,
        name: pack.name,
        version: pack.version,
        cdnUrl: pack.cdn_url,
        workerUrl: pack.worker_url,
        isPublic: pack.is_public,
        createdAt: pack.created_at,
        updatedAt: pack.updated_at,
        lastAccessed: pack.last_accessed,
        views: pack.views,
        downloads: pack.downloads,
        publisherId: pack.publisher_id,
        packageType: pack.package_type || 'basic',
        publishIp: pack.publish_ip,
        hasEncryption: !!pack.encrypted_key
    };

```

```

// Parse pack_json
try {
    const packJson = JSON.parse(pack.pack_json);
    enhancedPack.description = packJson.description || "";
    enhancedPack.author = packJson.author || "";
    enhancedPack.keywords = packJson.keywords || [];
    enhancedPack.homepage = packJson.homepage || "";
    enhancedPack.repository = packJson.repository || "";
    enhancedPack.license = packJson.license || "";
    enhancedPack.main = packJson.main || 'index.js';

```

```

    enhancedPack.scripts = packJson.scripts || {};
    enhancedPack.dependencies = packJson.dependencies || {};
    enhancedPack.devDependencies = packJson.devDependencies || {};
  } catch (e) {
    enhancedPack.description = "";
    enhancedPack.author = "";
    enhancedPack.keywords = [];
    enhancedPack.dependencies = {};
  }

  // Add metadata if available
  if (includeMetadata === 'true' && metadataMap[pack.id]) {
    const metadata = metadataMap[pack.id];
    enhancedPack.sandboxLevel = metadata.sandbox_level || 'basic';
    enhancedPack.requiresVerification = metadata.requires_verification || false;
    enhancedPack.verificationStatus = metadata.verification_status || 'unverified';
    enhancedPack.fileCount = metadata.file_count || 0;
    enhancedPack.totalSize = metadata.total_size || 0;

    // Calculate languages from files if available
    if (pack.files && typeof pack.files === 'object') {
      enhancedPack.languages = detectLanguagesFromFiles(pack.files);
    }
  }

  // Add version info
  if (versionsMap[pack.id] && versionsMap[pack.id].length > 0) {
    enhancedPack.versions = versionsMap[pack.id].map(v => ({
      version: v.version,
      versionNumber: v.version_number,
      checksum: v.checksum,
      createdAt: v.created_at,
      publisherId: v.publisher_id
    }));
    enhancedPack.latestVersion = versionsMap[pack.id][0].version;
    enhancedPack.versionCount = versionsMap[pack.id].length;
  }

  // Add collaborators
  if (collaboratorsMap[pack.id] && collaboratorsMap[pack.id].length > 0) {
    enhancedPack.collaborators = collaboratorsMap[pack.id].map(c => ({
      userId: c.user_id,
      permissionLevel: c.permission_level,
      invitedBy: c.invited_by,

```

```

        acceptedAt: c.accepted_at
    }));
    enhancedPack.collaboratorCount = collaboratorsMap[pack.id].length;
}

// Add dependencies
if (dependenciesMap[pack.id]) {
    enhancedPack.dependencyList = dependenciesMap[pack.id];
}

// Add recent changes
if (changesMap[pack.id] && changesMap[pack.id].length > 0) {
    enhancedPack.recentChanges = changesMap[pack.id].map(c => ({
        changeType: c.change_type,
        description: c.description,
        userId: c.user_id,
        createdAt: c.created_at,
        metadata: c.metadata
    }));
}

// Apply dependency filter - if dependency specified, only include packs that have it
if (dependency && useAdvancedSearch) {
    const hasDependency = dependenciesMap[pack.id] &&
        dependenciesMap[pack.id].some(dep =>
            dep.toLowerCase().includes(dependency.toLowerCase())
        );

    if (!hasDependency) {
        return null; // Skip this pack
    }
}

// Apply verification filter
if (verified === 'true' && useAdvancedSearch) {
    if (enhancedPack.verificationStatus !== 'verified') {
        return null; // Skip this pack
    }
}

// Calculate relevance score for search results
if (q) {
    enhancedPack.relevanceScore = calculateRelevanceScore(enhancedPack, q);
}

```

```

    return enhancedPack;
  }).filter(pack => pack !== null); // Remove null packs from filtering

// Apply relevance sorting if search query exists
let sortedPacks = enhancedPacks;
if (q && sort === 'relevance') {
  sortedPacks = enhancedPacks.sort((a, b) => {
    const scoreA = a.relevanceScore || 0;
    const scoreB = b.relevanceScore || 0;
    return scoreB - scoreA;
  });
}

// Build response
const response = {
  success: true,
  packs: sortedPacks,
  pagination: {
    page: pageNum,
    limit: limitNum,
    total: count || 0,
    totalPages: Math.ceil((count || 0) / limitNum),
    hasNextPage: (offset + limitNum) < (count || 0),
    hasPreviousPage: pageNum > 1
  },
  filters: {
    applied: {
      q: q || null,
      type: type || null,
      language: language || null,
      sort: sort || 'relevance',
      minDownloads: parseInt(minDownloads) || null,
      minViews: parseInt(minViews) || null,
      verified: verified === 'true' || null,
      hasReadme: hasReadme === 'true' || null,
      hasLicense: hasLicense === 'true' || null,
      hasTests: hasTests === 'true' || null,
      dependency: dependency || null,
      author: author || null
    },
    available: AVAILABLE_FILTERS
  },
  metadata: {

```

```

        processingTime: Date.now(),
        advancedSearch: useAdvancedSearch,
        includeMetadata: includeMetadata === 'true',
        totalPacks: count || 0,
        filteredPacks: sortedPacks.length
    }
};

res.status(200).json(response);

} catch (error) {
    console.error('Search error:', error);

    // Provide more specific error messages
    let statusCode = 500;
    let errorMessage = 'Search failed';
    let errorCode = 'SEARCH_ERROR';

    if (error.message.includes('does not exist')) {
        statusCode = 400;
        errorMessage = 'Database schema mismatch. Please check if all required tables exist.';
        errorCode = 'SCHEMA_MISMATCH';
    } else if (error.message.includes('timeout')) {
        statusCode = 504;
        errorMessage = 'Search timeout. Try simplifying your query.';
        errorCode = 'TIMEOUT';
    } else if (error.message.includes('rate limit')) {
        statusCode = 429;
        errorMessage = 'Rate limit exceeded. Please try again later.';
        errorCode = 'RATE_LIMIT';
    }

    res.status(statusCode).json({
        success: false,
        error: errorMessage,
        code: errorCode,
        details: process.env.NODE_ENV === 'development' ? error.message : undefined
    });
}

// Helper function to calculate relevance score
function calculateRelevanceScore(pack, searchQuery) {
    let score = 0;

```

```
const query = searchQuery.toLowerCase();

// Name match (highest weight)
if (pack.name.toLowerCase().includes(query)) {
  score += SEARCH_WEIGHTS.NAME_MATCH;
}

// Description match
if (pack.description && pack.description.toLowerCase().includes(query)) {
  score += SEARCH_WEIGHTS.DESCRPTION_MATCH;
}

// Keyword match
if (pack.keywords && Array.isArray(pack.keywords)) {
  const keywordMatches = pack.keywords.filter(keyword =>
    keyword.toLowerCase().includes(query)
  ).length;
  score += keywordMatches * SEARCH_WEIGHTS.KEYWORD_MATCH;
}

// Author match
if (pack.author && pack.author.toLowerCase().includes(query)) {
  score += SEARCH_WEIGHTS.DESCRPTION_MATCH;
}

// Popularity boost
score += Math.log10(pack.downloads + 1) * SEARCH_WEIGHTS.POPULARITY;

// Recency boost (packages from last 30 days get a boost)
const createdAt = new Date(pack.createdAt);
const daysOld = (Date.now() - createdAt.getTime()) / (1000 * 60 * 60 * 24);
if (daysOld < 30) {
  score += SEARCH_WEIGHTS.RECENCY * (30 - daysOld) / 30;
}

// Verification boost
if (pack.verificationStatus === 'verified') {
  score += 5;
}

// Recent activity boost
if (pack.recentChanges && pack.recentChanges.length > 0) {
  const latestChange = new Date(pack.recentChanges[0].createdAt);
  const daysSinceChange = (Date.now() - latestChange.getTime()) / (1000 * 60 * 60 * 24);
```

```
    if (daysSinceChange < 7) {
      score += 3;
    }
  }

  return score;
}

// Helper function to detect languages from files
function detectLanguagesFromFiles(files) {
  const languages = new Set();
  const extMap = {
    'js': 'javascript',
    'mjs': 'javascript',
    'cjs': 'javascript',
    'jsx': 'javascript',
    'ts': 'javascript',
    'tsx': 'javascript',
    'py': 'python',
    'pyc': 'python',
    'pyo': 'python',
    'wasm': 'wasm',
    'json': 'json',
    'md': 'markdown',
    'markdown': 'markdown',
    'txt': 'text',
    'html': 'html',
    'htm': 'html',
    'css': 'css',
    'scss': 'css',
    'sass': 'css',
    'png': 'image',
    'jpg': 'image',
    'jpeg': 'image',
    'gif': 'image',
    'svg': 'image',
    'webp': 'image',
    'csv': 'data',
    'tsv': 'data',
    'xml': 'data',
    'yaml': 'data',
    'yml': 'data'
  };
};
```

```
Object.keys(files).forEach(filename => {
  const ext = filename.split('.').pop().toLowerCase();
  if (extMap[ext]) {
    languages.add(extMap[ext]);
  }
});

if (languages.size === 0) {
  languages.add('javascript'); // Default
}

return Array.from(languages);
}

// Helper function to extract description from pack_json
function extractDescription(packJson) {
  try {
    const json = typeof packJson === 'string' ? JSON.parse(packJson) : packJson;
    return json.description || "";
  } catch (e) {
    return "";
  }
}

// Helper function to extract author from pack_json
function extractAuthor(packJson) {
  try {
    const json = typeof packJson === 'string' ? JSON.parse(packJson) : packJson;
    return json.author || "";
  } catch (e) {
    return "";
  }
}

// Export for testing
if (process.env.NODE_ENV === 'test') {
  module.exports = {
    calculateRelevanceScore,
    detectLanguagesFromFiles,
    extractDescription,
    extractAuthor
  };
}
```

```

api/admin/database.js
// /api/admin/database.js - FIXED WITH MASKED URL

import { createClient } from '@supabase/supabase-js'
import rateLimit from 'express-rate-limit'
import helmet from 'helmet'
import crypto from 'crypto'

// Security middleware wrapper
const withSecurity = (handler) => async (req, res) => {
  try {
    // Apply Helmet security headers
    await new Promise((resolve) => {
      helmet({
        contentSecurityPolicy: {
          directives: {
            defaultSrc: ["'self'"],
            styleSrc: ["'self'", "'unsafe-inline'"],
            scriptSrc: ["'self'"],
            imgSrc: ["'self'", "data:", "https:"],
          },
        },
        hsts: {
          maxAge: 31536000,
          includeSubDomains: true,
          preload: true
        }
      })(req, res, resolve)
    })

    // 1. Get environment variables
    const ADMIN_PASSWORD = process.env.ADMIN_PASSWORD
    const SUPABASE_URL = process.env.SUPABASE_URL
    const SUPABASE_SERVICE_ROLE_KEY =
process.env.SUPABASE_SERVICE_ROLE_KEY

    // Verify all required secrets are present
    if (!ADMIN_PASSWORD || !SUPABASE_URL || !SUPABASE_SERVICE_ROLE_KEY) {
      console.error('✗ Missing required environment variables')
      return res.status(500).json({
        error: 'Server configuration error',
        firewall: 'active'
      })
    }
  }
}

```

```

// 2. IP-based firewall with hashed validation
const clientIP = req.headers['x-forwarded-for'] || req.socket.remoteAddress || req.ip

// Get timestamp from request
const timestamp = req.body?.timestamp || ''

// Parse IP validation from request headers
const requestIPHash = req.headers['x-ip-validator']

// Generate expected hash: SHA256(ADMIN_PASSWORD + timestamp.slice(0, 10))
const expectedIPHash = crypto
  .createHash('sha256')
  .update(ADMIN_PASSWORD + timestamp.slice(0, 10))
  .digest('hex')
  .slice(0, 32)

console.log('🔒 IP VALIDATOR DEBUG:', {
  received: requestIPHash?.substring(0, 8) + '...',
  expected: expectedIPHash.substring(0, 8) + '...',
  password: ADMIN_PASSWORD?.substring(0, 3) + '...',
  timestampSlice: timestamp.slice(0, 10),
  formula: 'SHA256(password + timestamp.slice(0, 10))'
})

// Enhanced IP validation
if (!requestIPHash || requestIPHash !== expectedIPHash) {
  console.warn('🚫 Firewall blocked: Invalid IP validator from ${clientIP}')
  return res.status(403).json({
    error: 'Firewall violation - Invalid request signature',
    timestamp: new Date().toISOString(),
    firewall: 'high-security',
    hint: 'IP validator must be SHA256(password + YYYY-MM-DD)'
  })
}

// 3. Rate Limiting with express-rate-limit
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 10, // 10 requests per window
  message: {
    error: 'Rate limit exceeded',
    firewall: 'active',
    retryAfter: '15 minutes'
  }
})

```

```

    },
    standardHeaders: true,
    legacyHeaders: false,
    keyGenerator: (req) => {
      // Use IP + user agent for rate limiting
      const userAgent = req.headers['user-agent'] || 'unknown'
      return `${clientIP}-${crypto.createHash('md5').update(userAgent).digest('hex')}`
    },
    handler: (req, res) => {
      console.warn(`🔥 Rate limit exceeded for IP: ${clientIP}`)
      res.status(429).json({
        error: 'Too many requests',
        firewall: 'rate-limit',
        retryAfter: '15 minutes'
      })
    }
  })
})

```

```

// Apply rate limiting
await new Promise((resolve, reject) => {
  limiter(req, res, (result) => {
    if (result instanceof Error) reject(result)
    resolve(result)
  })
})

```

```

// 4. Request validation
if (req.method !== 'POST') {
  return res.status(405).json({
    error: 'Method not allowed',
    allowedMethods: ['POST']
  })
}

```

```

const { password, timestamp: bodyTimestamp, signature, action } = req.body

```

```

if (!password || !bodyTimestamp || !signature || !action) {
  return res.status(400).json({
    error: 'Missing required fields',
    required: ['password', 'timestamp', 'signature', 'action']
  })
}

```

```

// 5. Timestamp validation (prevent replay attacks)

```

```

const requestTime = new Date(bodyTimestamp).getTime()
const currentTime = Date.now()
const timeDiff = Math.abs(currentTime - requestTime)

if (isNaN(requestTime) || timeDiff > 5 * 60 * 1000) { // 5 minute window
  return res.status(401).json({
    error: 'Request expired or invalid timestamp',
    maxWindow: '5 minutes'
  })
}

// 6. PASSWORD VALIDATION (must come before signature check)
if (password !== ADMIN_PASSWORD) {
  console.warn('🚨 Failed password attempt from IP: ${clientIP}')
  return res.status(401).json({
    error: 'Unauthorized access',
    firewall: 'password-validation'
  })
}

// 7. SIGNATURE VALIDATION - FIXED WITH MASKED URL
// Get masked URL for signature (EXACTLY matches frontend)
const maskedSupabaseUrl = SUPABASE_URL.replace(/\/(.*)\.supabase\.co/,
'/**.supabase.co');

// Start with the base message that BOTH frontend and backend agree on
let signatureMessage = `${password}:${bodyTimestamp}:${action}:${maskedSupabaseUrl}`;

// Get the data from request body
const bodyData = req.body.data;

console.log('🔓 SIGNATURE DEBUG - START:', {
  action: action,
  hasDataInBody: 'data' in req.body,
  dataType: typeof bodyData,
  dataValue: bodyData,
  maskedUrl: maskedSupabaseUrl
});

// CRITICAL FIX: Handle data inclusion EXACTLY like frontend
if (!['GET_ALL_TABLES', 'GET_TABLE'].includes(action)) {
  // For these actions, frontend passes data = null, so signature should NOT include data
  console.log('🔓 SKIPPING data for read-only actions');
}

```

```

// For other actions (INSERT, UPDATE, DELETE, RAW_SQL)
else if (bodyData !== null && bodyData !== undefined) {
  // Check if it's an object with keys
  if (typeof bodyData === 'object' && Object.keys(bodyData).length > 0) {
    const dataString = JSON.stringify(bodyData);
    signatureMessage += `:${dataString}`;
    console.log('🔒 INCLUDING object data:', dataString.substring(0, 50) + '...');
  }
  // Check if it's a non-empty string
  else if (typeof bodyData === 'string' && bodyData.trim() !== "") {
    signatureMessage += `:${bodyData}`;
    console.log('🔒 INCLUDING string data:', bodyData.substring(0, 50) + '...');
  }
  else {
    console.log('🔒 NO data to include (empty or invalid)');
  }
}
else {
  console.log('🔒 NO data field or data is null/undefined');
}

console.log('🔒 BACKEND FINAL SIGNATURE MESSAGE:', signatureMessage.substring(0, 100) + '...');

// IMPORTANT: Use password as HMAC key (matches frontend)
const expectedSignature = crypto
  .createHmac('sha256', password)
  .update(signatureMessage)
  .digest('hex');

console.log('🔒 SIGNATURE COMPARISON:', {
  received: signature?.substring(0, 16) + '...',
  expected: expectedSignature?.substring(0, 16) + '...',
  matches: signature === expectedSignature,
  receivedFull: signature,
  expectedFull: expectedSignature
});

if (signature !== expectedSignature) {
  console.warn('🚨 Invalid signature from IP: ${clientIP}');
  console.warn('Expected message was:', signatureMessage);
  console.warn('Full expected signature:', expectedSignature);
  console.warn('Full received signature:', signature);
  return res.status(401).json({

```

```

        error: 'Invalid request signature',
        firewall: 'signature-validation',
        details: 'Signature mismatch - check URL format'
    });
}

// 8. Create Supabase client with service role key
const supabase = createClient(
  SUPABASE_URL,
  SUPABASE_SERVICE_ROLE_KEY,
  {
    auth: {
      autoRefreshToken: false,
      persistSession: false,
      detectSessionInUrl: false
    },
    global: {
      headers: {
        'X-Client-IP': clientIP,
        'X-Admin-Access': 'true'
      }
    }
  }
)

// 9. All security checks passed - proceed to handler
console.log(`✅ Secure access granted for action: ${action} from IP: ${clientIP}`);
return handler(req, res, supabase, clientIP);

} catch (error) {
  console.error('❌ Security middleware error:', error);
  return res.status(500).json({
    error: 'Security check failed',
    details: 'Internal server error',
    firewall: 'error'
  });
}
}

// Main handler with full database management
export default withSecurity(async (req, res, supabase, clientIP) => {
  try {
    const { action, table, data, query, id, limit = 1000 } = req.body

```

```

// Additional action validation
const validActions = ['GET_ALL_TABLES', 'GET_TABLE', 'INSERT', 'UPDATE', 'DELETE',
'RAW_SQL', 'EXECUTE_FUNCTION']
if (!validActions.includes(action)) {
  return res.status(400).json({
    error: 'Invalid action',
    validActions: validActions
  })
}

switch (action) {
case 'GET_ALL_TABLES':
  try {
    // Use the RPC function to get all tables
    const { data: tables, error: tablesError } = await supabase
      .rpc('get_all_tables')

    if (tablesError) {
      console.error('Tables fetch error (RPC):', tablesError)

      // Fallback: Try to get tables from pg_tables via SQL
      const { data: pgTables, error: pgError } = await supabase
        .rpc('exec_sql', {
          query_text: `SELECT tablename as table_name, 'BASE TABLE' as table_type FROM
pg_catalog.pg_tables WHERE schemaname = 'public' ORDER BY tablename`,
          is_readonly: true
        }).catch(err => ({ error: err }))

      if (pgError) {
        console.error('All table fetch methods failed:', pgError)
        return res.status(500).json({
          error: 'Failed to fetch tables',
          details: 'Cannot access database schema.',
          hint: 'Make sure get_all_tables() function exists and SERVICE_ROLE_KEY has
proper permissions'
        })
      }

      // Use pg_catalog result
      tables = pgTables
    }

    console.log(`Found ${tables.length} tables in database`)

```

```

// Fetch sample data from each table
const allTablesData = {}
const tableStats = {}

// Limit concurrent table checks to avoid rate limits
const maxConcurrent = 3
const tableChunks = []
for (let i = 0; i < tables.length; i += maxConcurrent) {
  tableChunks.push(tables.slice(i, i + maxConcurrent))
}

for (const chunk of tableChunks) {
  await Promise.all(chunk.map(async (tableInfo) => {
    const tableName = tableInfo.table_name || tableInfo.tablename

    if (!tableName) return

    try {
      const { data: tableData, error: tableError, count } = await supabase
        .from(tableName)
        .select('*', { count: 'exact', head: true })
        .limit(1)

      if (!tableError) {
        tableStats[tableName] = {
          type: tableInfo.table_type || 'BASE TABLE',
          estimatedRows: count || 0,
          sampled: tableData ? tableData.length : 0
        }

        // Get full data for smaller tables
        if (count && count <= limit) {
          const { data: fullData } = await supabase
            .from(tableName)
            .select('*')
            .limit(limit)
          allTablesData[tableName] = fullData || []
        }
      } else {
        console.warn(` Could not fetch table ${tableName}:`, tableError.message)
        tableStats[tableName] = {
          type: tableInfo.table_type || 'BASE TABLE',
          estimatedRows: 0,
          sampled: 0,

```

```

        error: tableError.message
      }
    }
  } catch (err) {
    console.warn(`Error fetching table ${tableName}:`, err.message)
    tableStats[tableName] = {
      type: tableInfo.table_type || 'BASE TABLE',
      estimatedRows: 0,
      sampled: 0,
      error: err.message
    }
  }
}
)))

// Small delay between chunks
if (tableChunks.length > 1) {
  await new Promise(resolve => setTimeout(resolve, 100))
}
}

return res.json({
  success: true,
  timestamp: new Date().toISOString(),
  ip: clientIP,
  tables: tables.map(t => t.table_name || t.tablename).filter(Boolean),
  stats: tableStats,
  data: allTablesData,
  limit: limit,
  supabaseUrl: process.env.SUPABASE_URL?.replace(/\/(.*?)\.supabase.co/,
'/***.supabase.co')
})

} catch (error) {
  console.error('Table fetch error:', error)
  return res.status(500).json({
    error: 'Failed to fetch tables',
    details: error.message
  })
}

case 'GET_TABLE':
  if (!table) {
    return res.status(400).json({ error: 'Table name required' })
  }

```

```

const { data: tableData, error: tableError, count } = await supabase
  .from(table)
  .select('*', { count: 'exact' })
  .limit(limit)

if (tableError) {
  console.error(`Table ${table} fetch error:`, tableError)
  return res.status(500).json({
    error: `Failed to fetch table: ${table}`,
    details: tableError.message
  })
}

return res.json({
  success: true,
  table: table,
  count: count || 0,
  data: tableData,
  limit: limit,
  ip: clientIP
})

case 'INSERT':
  if (!table || !data) {
    return res.status(400).json({ error: 'Table and data required' })
  }

  // Validate data structure
  if (!Array.isArray(data)) {
    return res.status(400).json({ error: 'Data must be an array' })
  }

  const { data: inserted, error: insertError } = await supabase
    .from(table)
    .insert(data)
    .select()

  if (insertError) {
    console.error(`Insert error for table ${table}:`, insertError)
    return res.status(500).json({
      error: 'Insert failed',
      details: insertError.message,
      table: table
    })
  }

```

```
  })  
}
```

```
console.log(`✅ Inserted ${inserted.length} rows into ${table} from IP: ${clientIP}`)  
return res.json({  
  success: true,  
  action: 'insert',  
  table: table,  
  insertedCount: inserted.length,  
  data: inserted,  
  ip: clientIP  
})
```

```
case 'UPDATE':  
  if (!table || !id || !data) {  
    return res.status(400).json({ error: 'Table, id, and data required' })  
  }
```

```
const { data: updated, error: updateError } = await supabase  
  .from(table)  
  .update(data)  
  .eq('id', id)  
  .select()
```

```
if (updateError) {  
  console.error(`Update error for table ${table}, id ${id}:`, updateError)  
  return res.status(500).json({  
    error: 'Update failed',  
    details: updateError.message,  
    table: table,  
    id: id  
  })  
}
```

```
console.log(`✅ Updated row ${id} in ${table} from IP: ${clientIP}`)  
return res.json({  
  success: true,  
  action: 'update',  
  table: table,  
  id: id,  
  data: updated,  
  ip: clientIP  
})
```

```

case 'DELETE':
  if (!table || !id) {
    return res.status(400).json({ error: 'Table and id required' })
  }

  const { data: deleted, error: deleteError } = await supabase
    .from(table)
    .delete()
    .eq('id', id)
    .select()

  if (deleteError) {
    console.error(`Delete error for table ${table}, id ${id}:`, deleteError)
    return res.status(500).json({
      error: 'Delete failed',
      details: deleteError.message,
      table: table,
      id: id
    })
  }

  console.log(`✅ Deleted row ${id} from ${table} from IP: ${clientIP}`)
  return res.json({
    success: true,
    action: 'delete',
    table: table,
    id: id,
    data: deleted,
    ip: clientIP
  })

case 'RAW_SQL':
  if (!query) {
    return res.status(400).json({ error: 'SQL query required' })
  }

  // For security, we'll use rpc for SQL execution
  const { data: sqlResult, error: sqlError } = await supabase.rpc('exec_sql', {
    query_text: query,
    is_readonly: query.trim().toLowerCase().startsWith('select')
  }).catch(err => ({ error: err }))

  if (sqlError) {
    console.error(`SQL execution error:`, sqlError)
  }

```

```

        return res.status(500).json({
          error: 'SQL execution failed',
          details: sqlError.message,
          query: query.substring(0, 100) + '...'
        })
      }

      console.log(`✅ SQL executed from IP: ${clientIP}`)
      return res.json({
        success: true,
        action: 'raw_sql',
        result: sqlResult,
        ip: clientIP
      })

      default:
        return res.status(400).json({
          error: 'Invalid action',
          validActions: ['GET_ALL_TABLES', 'GET_TABLE', 'INSERT', 'UPDATE', 'DELETE',
'RAW_SQL']
        })
      }
    } catch (error) {
      console.error('❌ Database operation error:', error)
      return res.status(500).json({
        error: 'Database operation failed',
        details: error.message,
        ip: clientIP,
        timestamp: new Date().toISOString()
      })
    }
  })
})

api/Old/search.js
// /api/search.js
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(
  process.env.SUPABASE_URL,
  process.env.SUPABASE_ANON_KEY
);

```

```

export default async function handler(req, res) {
  const { q, type, page = 1, limit = 20 } = req.query;

  try {
    let query = supabase
      .from('packs')
      .select('id, name, version, package_type, views, downloads, created_at')
      .eq('is_public', true)
      .order('created_at', { ascending: false });

    // Apply filters
    if (q) {
      query = query.ilike('name', `%${q}%`);
    }
    if (type) {
      query = query.eq('package_type', type);
    }

    // Pagination
    const from = (page - 1) * limit;
    const to = from + limit - 1;
    query = query.range(from, to);

    const { data, error, count } = await query;

    if (error) throw error;

    res.status(200).json({
      success: true,
      packs: data || [],
      page: parseInt(page),
      total: count,
      hasMore: data?.length === limit
    });
  } catch (error) {
    res.status(500).json({ error: 'Search failed' });
  }
}

```

```

HTML_FILES
public/Docs/docs.html
<!DOCTYPE html>

```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PackCDN - Interactive Documentation</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #0a0a14 0%, #1a1a2e 50%, #0f3460 100%);
      color: #ffffff;
      min-height: 100vh;
    }

    /* Animated background */
    .background-animation {
      position: fixed;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      z-index: -1;
      overflow: hidden;
    }

    .gradient-circle {
      position: absolute;
      border-radius: 50%;
      filter: blur(40px);
      opacity: 0.3;
      animation: float 20s infinite ease-in-out;
    }

    .circle-1 {
      width: 300px;
      height: 300px;
      background: linear-gradient(45deg, #667eea, #764ba2);
      top: 10%;
      left: 10%;
```

```
    animation-delay: 0s;
}

.circle-2 {
    width: 400px;
    height: 400px;
    background: linear-gradient(45deg, #64ffda, #00b4d8);
    top: 60%;
    right: 10%;
    animation-delay: -5s;
}

.circle-3 {
    width: 250px;
    height: 250px;
    background: linear-gradient(45deg, #ff6b6b, #ffd93d);
    bottom: 10%;
    left: 20%;
    animation-delay: -10s;
}

@keyframes float {
    0%, 100% { transform: translateY(0) scale(1); }
    33% { transform: translateY(-20px) scale(1.1); }
    66% { transform: translateY(20px) scale(0.9); }
}

.nav-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    padding: 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    background: rgba(10, 10, 20, 0.95);
    backdrop-filter: blur(10px);
    border-bottom: 1px solid rgba(102, 126, 234, 0.3);
    z-index: 1000;
}

.logo {
    font-size: 24px;
```

```
font-weight: bold;
background: linear-gradient(45deg, #667eea, #764ba2);
-webkit-background-clip: text;
-webkit-text-fill-color: transparent;
}

.nav-links {
  display: flex;
  gap: 30px;
}

.nav-links a {
  color: #ffffff;
  text-decoration: none;
  padding: 8px 16px;
  border-radius: 20px;
  transition: all 0.3s ease;
  font-weight: 500;
}

.nav-links a:hover {
  background: rgba(102, 126, 234, 0.3);
  transform: translateY(-2px);
}

.content {
  max-width: 1200px;
  margin: 0 auto;
  padding: 140px 40px 60px;
}

.hero-section {
  text-align: center;
  padding: 80px 0;
  margin-bottom: 80px;
  opacity: 0;
  animation: fadeInUp 1s forwards;
}

.hero-title {
  font-size: 4rem;
  margin-bottom: 20px;
  background: linear-gradient(45deg, #667eea, #764ba2);
  -webkit-background-clip: text;
```

```
-webkit-text-fill-color: transparent;  
}
```

```
.hero-subtitle {  
  font-size: 1.5rem;  
  color: #ffffff;  
  margin-bottom: 40px;  
  max-width: 600px;  
  margin-left: auto;  
  margin-right: auto;  
  opacity: 0.9;  
  line-height: 1.6;  
}
```

```
.cta-buttons {  
  display: flex;  
  gap: 20px;  
  justify-content: center;  
}
```

```
.btn {  
  padding: 15px 30px;  
  border-radius: 50px;  
  text-decoration: none;  
  font-weight: 600;  
  font-size: 1rem;  
  transition: all 0.3s ease;  
  cursor: pointer;  
  display: inline-block;  
  text-align: center;  
  color: white;  
}
```

```
.btn-primary {  
  background: linear-gradient(45deg, #667eea, #764ba2);  
  color: white;  
  border: none;  
}
```

```
.btn-secondary {  
  background: transparent;  
  color: #ffffff;  
  border: 2px solid #667eea;  
  background: rgba(102, 126, 234, 0.1);
```

```
}

.btn:hover {
  transform: translateY(-3px);
  box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);
}

.section {
  margin: 100px 0;
  padding: 50px;
  background: rgba(25, 25, 35, 0.85);
  border-radius: 20px;
  border: 1px solid rgba(102, 126, 234, 0.3);
  backdrop-filter: blur(10px);
  opacity: 0;
  transform: translateY(40px);
  animation: fadeInUp 0.8s forwards;
}

.section.visible {
  opacity: 1;
  transform: translateY(0);
}

.section-title {
  font-size: 2.5rem;
  margin-bottom: 30px;
  color: #ffffff;
  display: flex;
  align-items: center;
  gap: 15px;
}

.section-title .icon {
  font-size: 2rem;
  color: #667eea;
}

.cards-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
  gap: 30px;
  margin: 30px 0;
}
```

```
.card {
  background: rgba(35, 35, 50, 0.9);
  padding: 30px;
  border-radius: 15px;
  transition: all 0.3s ease;
  border: 1px solid rgba(102, 126, 234, 0.2);
}

.card:hover {
  transform: translateY(-10px);
  border-color: #667eea;
  box-shadow: 0 15px 30px rgba(102, 126, 234, 0.2);
}

.card-title {
  font-size: 1.5rem;
  margin-bottom: 15px;
  color: #ffffff;
  font-weight: 600;
}

.card-content {
  color: #ffffff;
  line-height: 1.6;
  opacity: 0.9;
}

.code-block {
  background: rgba(15, 15, 25, 0.95);
  padding: 25px;
  border-radius: 10px;
  margin: 20px 0;
  overflow-x: auto;
  font-family: 'Courier New', monospace;
  border-left: 4px solid #667eea;
  color: #e2e8f0;
  font-size: 0.95rem;
}

.code-block .highlight {
  color: #64ffda;
  font-weight: bold;
}
```

```
.terminal {
  background: rgba(10, 10, 20, 0.95);
  border: 2px solid #667eea;
  border-radius: 10px;
  overflow: hidden;
  margin: 30px 0;
}

.terminal-header {
  background: linear-gradient(45deg, #667eea, #764ba2);
  padding: 10px 20px;
  display: flex;
  align-items: center;
  gap: 10px;
}

.terminal-dot {
  width: 12px;
  height: 12px;
  border-radius: 50%;
  background: rgba(255, 255, 255, 0.7);
}

.terminal-content {
  padding: 20px;
  font-family: 'Courier New', monospace;
  color: #00ff9d;
  background: rgba(0, 0, 0, 0.5);
  line-height: 1.5;
}

.diagram-container {
  position: relative;
  height: 400px;
  margin: 40px 0;
  background: rgba(20, 20, 35, 0.8);
  border-radius: 10px;
  overflow: hidden;
  border: 1px solid rgba(102, 126, 234, 0.3);
}

.flow-node {
  position: absolute;
```

```
width: 150px;
padding: 20px;
background: rgba(102, 126, 234, 0.3);
border: 2px solid #667eea;
border-radius: 10px;
text-align: center;
transition: all 0.3s ease;
color: white;
font-weight: 600;
backdrop-filter: blur(5px);
animation: floatNode 3s infinite ease-in-out;
}

.flow-node:hover {
  background: rgba(102, 126, 234, 0.5);
  transform: scale(1.05);
}

.flow-line {
  position: absolute;
  height: 2px;
  background: linear-gradient(90deg, #667eea, #764ba2);
  transform-origin: left center;
}

.feature-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
  margin-top: 30px;
}

.feature-item {
  display: flex;
  align-items: center;
  gap: 15px;
  padding: 25px;
  background: rgba(40, 40, 50, 0.85);
  border-radius: 10px;
  transition: all 0.3s ease;
  border: 1px solid rgba(102, 126, 234, 0.2);
}

.feature-item:hover {
```

```
    background: rgba(102, 126, 234, 0.3);
    transform: translateX(10px);
    border-color: #667eea;
}

.feature-icon {
    font-size: 2rem;
    color: #667eea;
}

.feature-item h3 {
    color: #ffffff;
    margin-bottom: 5px;
}

.feature-item p {
    color: #ffffff;
    opacity: 0.9;
}

.stats-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 20px;
    margin: 40px 0;
}

.stat-box {
    text-align: center;
    padding: 30px;
    background: rgba(102, 126, 234, 0.2);
    border-radius: 15px;
    border: 1px solid rgba(102, 126, 234, 0.4);
    backdrop-filter: blur(5px);
    animation: pulse 2s infinite;
}

.stat-number {
    font-size: 3rem;
    font-weight: bold;
    color: #ffffff;
    margin-bottom: 10px;
}
```

```
.stat-label {
  color: #ffffff;
  font-size: 1.1rem;
  opacity: 0.9;
}

.interactive-demo {
  position: relative;
  height: 300px;
  margin: 40px 0;
  border: 2px solid rgba(102, 126, 234, 0.4);
  border-radius: 15px;
  overflow: hidden;
  background: rgba(15, 15, 25, 0.9);
  display: flex;
  align-items: center;
  justify-content: center;
}

.demo-cube {
  width: 100px;
  height: 100px;
  background: linear-gradient(45deg, #667eea, #764ba2);
  border-radius: 10px;
  animation: rotateCube 10s linear infinite;
  box-shadow: 0 10px 30px rgba(102, 126, 234, 0.5);
}

.demo-controls {
  display: flex;
  gap: 10px;
  justify-content: center;
  margin-top: 20px;
}

.demo-btn {
  padding: 12px 24px;
  background: linear-gradient(45deg, #667eea, #764ba2);
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s ease;
  font-weight: 600;
}
```

```
}

.demo-btn:hover {
  background: linear-gradient(45deg, #764ba2, #667eea);
  transform: scale(1.1);
}

footer {
  text-align: center;
  padding: 50px 40px;
  margin-top: 100px;
  border-top: 2px solid rgba(102, 126, 234, 0.3);
  color: #ffffff;
  background: rgba(10, 10, 20, 0.95);
}

footer a {
  color: #667eea;
  text-decoration: none;
  transition: all 0.3s ease;
  font-weight: 500;
}

footer a:hover {
  color: #764ba2;
  text-decoration: underline;
}

.back-to-top {
  position: fixed;
  bottom: 30px;
  right: 30px;
  width: 50px;
  height: 50px;
  background: linear-gradient(45deg, #667eea, #764ba2);
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: pointer;
  opacity: 0;
  transition: all 0.3s ease;
  color: white;
  font-size: 1.2rem;
```

```
    z-index: 1000;
}

.back-to-top.visible {
    opacity: 1;
    transform: translateY(0);
}

.back-to-top:hover {
    background: linear-gradient(45deg, #764ba2, #667eea);
    transform: translateY(-5px) scale(1.1);
}

/* Animations */
@keyframes fadeInUp {
    from {
        opacity: 0;
        transform: translateY(40px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

@keyframes floatNode {
    0%, 100% { transform: translateY(0); }
    50% { transform: translateY(-10px); }
}

@keyframes pulse {
    0%, 100% { transform: scale(1); }
    50% { transform: scale(1.05); }
}

@keyframes rotateCube {
    0% { transform: rotateX(0) rotateY(0) rotateZ(0); }
    100% { transform: rotateX(360deg) rotateY(360deg) rotateZ(360deg); }
}

@media (max-width: 768px) {
    .hero-title {
        font-size: 2.5rem;
    }
}
```

```
.nav-links {
  display: none;
}

.content {
  padding: 140px 20px 20px;
}

.section {
  padding: 30px 20px;
  margin: 50px 0;
}

.cards-container {
  grid-template-columns: 1fr;
}
}
</style>
</head>
<body>
  <!-- Simple background animation -->
  <div class="background-animation">
    <div class="gradient-circle circle-1"></div>
    <div class="gradient-circle circle-2"></div>
    <div class="gradient-circle circle-3"></div>
  </div>

  <!-- Navigation -->
  <nav class="nav-overlay">
    <div class="logo">PackCDN</div>
    <div class="nav-links">
      <a href="#overview">Overview</a>
      <a href="#features">Features</a>
      <a href="#api">API</a>
      <a href="#architecture">Architecture</a>
      <a href="#getting-started">Get Started</a>
      <a href="selector.html">Switch View</a>
    </div>
  </nav>

  <!-- Main content -->
  <div class="content">
    <section class="hero-section">
```

```
<h1 class="hero-title">PackCDN Documentation</h1>
<p class="hero-subtitle">
  Instant package publishing and distribution platform with global CDN,
  built on Cloudflare Workers and Vercel Serverless Functions.
</p>
<div class="cta-buttons">
  <a href="#getting-started" class="btn btn-primary">Quick Start</a>
  <a href="editor.html" class="btn btn-secondary">Try Editor</a>
</div>
</section>

<section id="overview" class="section">
  <h2 class="section-title"><span class="icon">📦</span> Overview</h2>
  <p class="card-content">
    PackCDN is a zero-configuration platform for instantly publishing and sharing code
    packages.
    No registration required. Anonymous and free forever.
  </p>

  <div class="stats-container">
    <div class="stat-box">
      <div class="stat-number">4</div>
      <div class="stat-label">API Endpoints</div>
    </div>
    <div class="stat-box">
      <div class="stat-number">3</div>
      <div class="stat-label">Package Types</div>
    </div>
    <div class="stat-box">
      <div class="stat-number">∞</div>
      <div class="stat-label">Global CDN</div>
    </div>
    <div class="stat-box">
      <div class="stat-number">0</div>
      <div class="stat-label">Registration</div>
    </div>
  </div>
</section>

<section id="features" class="section">
  <h2 class="section-title"><span class="icon">✨</span> Features</h2>

  <div class="feature-grid">
    <div class="feature-item">
```

```
<span class="feature-icon">🚀</span>
<div>
  <h3>Instant Publishing</h3>
  <p>Upload code, get CDN URLs in seconds</p>
</div>
</div>
<div class="feature-item">
  <span class="feature-icon">🌐</span>
  <div>
    <h3>Global CDN</h3>
    <p>Cloudflare Workers edge network</p>
  </div>
</div>
<div class="feature-item">
  <span class="feature-icon">🔒</span>
  <div>
    <h3>Anonymous</h3>
    <p>No accounts, no tracking</p>
  </div>
</div>
<div class="feature-item">
  <span class="feature-icon">⚡</span>
  <div>
    <h3>WASM Support</h3>
    <p>Upload and serve WebAssembly</p>
  </div>
</div>
<div class="feature-item">
  <span class="feature-icon">🔍</span>
  <div>
    <h3>Smart Analysis</h3>
    <p>Automatic dependency detection</p>
  </div>
</div>
<div class="feature-item">
  <span class="feature-icon">🔄</span>
  <div>
    <h3>Multi-format</h3>
    <p>JavaScript, Python, WASM</p>
  </div>
</div>
</div>
</section>
```

```
<section id="api" class="section">
  <h2 class="section-title"><span class="icon">🔔</span> API Reference</h2>

  <div class="cards-container">
    <div class="card">
      <h3 class="card-title">GET /api/get-pack</h3>
      <p class="card-content">Retrieve package by ID or URL ID</p>
      <div class="code-block">
        <span class="highlight">GET</span>
        https://pack-cdn.vercel.app/api/get-pack?id={id}
      </div>
    </div>

    <div class="card">
      <h3 class="card-title">POST /api/publish</h3>
      <p class="card-content">Publish new package with CDN</p>
      <div class="code-block">
        <span class="highlight">POST</span> /api/publish<br>
        { "name": "my-package", "files": {...} }
      </div>
    </div>

    <div class="card">
      <h3 class="card-title">POST /api/analyze</h3>
      <p class="card-content">Analyze code for dependencies</p>
      <div class="code-block">
        <span class="highlight">POST</span> /api/analyze<br>
        { "code": "import React", "type": "npm" }
      </div>
    </div>

    <div class="card">
      <h3 class="card-title">GET /api/search</h3>
      <p class="card-content">Search public packages</p>
      <div class="code-block">
        <span class="highlight">GET</span> /api/search?q=react&type=npm
      </div>
    </div>

    <div class="terminal">
      <div class="terminal-header">
        <div class="terminal-dot"></div>
        <div class="terminal-dot"></div>
```

```
<div class="terminal-dot"></div>
</div>
<div class="terminal-content">
  $ curl -X POST https://pack-cdn.vercel.app/api/publish \<br>
  -H "Content-Type: application/json" \<br>
  -d '{"name":"my-package","files":{"index.js":"..."}}'\<br><br>
  {\<br>
    "success": true,<br>
    "cdnUrl": "https://packcdn.../cdn/abc123",<br>
    "installCommand": "pack install my-package https://..."<br>
  }
</div>
</div>
</section>
```

```
<section id="architecture" class="section">
  <h2 class="section-title"><span class="icon">🏗️</span> Architecture</h2>
```

```
<div class="diagram-container" id="flow-diagram">
  <!-- Flow diagram will be created with JavaScript -->
</div>
```

```
<div class="code-block">
  // Package Storage Format in Supabase<br>
  {\<br>
    "url_id": "abc123xyz",<br>
    "name": "my-package",<br>
    "files": {\<br>
      "index.js": "export default function() {...}",<br>
      "module.wasm": "AGFzbQEAAAAG..."<br>
    },<br>
    "pack_json": "{\"type\":\"module\",\"version\":\"1.0.0\"}"<br>
  }
</div>
</section>
```

```
<section id="getting-started" class="section">
  <h2 class="section-title"><span class="icon">🚀</span> Getting Started</h2>
```

```
<div class="cards-container">
  <div class="card">
    <h3 class="card-title">1. Create Package</h3>
    <p class="card-content">Use our built-in editor or upload existing files</p>
  </div>
```

```
<div class="card">
  <h3 class="card-title">2. Configure</h3>
  <p class="card-content">Set package name, type, and visibility</p>
</div>

<div class="card">
  <h3 class="card-title">3. Publish</h3>
  <p class="card-content">Get instant CDN URLs and installation commands</p>
</div>

<div class="card">
  <h3 class="card-title">4. Use Anywhere</h3>
  <p class="card-content">Import directly in browser, Node.js, or other projects</p>
</div>

<div class="interactive-demo" id="demo-container">
  <div class="demo-cube" id="demoCube"></div>
</div>

<div class="demo-controls">
  <button class="demo-btn" onclick="runDemo()">Run Demo</button>
  <button class="demo-btn" onclick="resetDemo()">Reset</button>
</div>

<div class="code-block">
  // Using a published package<br>
  import myPackage from 'https://packcdn.../cdn/abc123';<br><br>
  // or in HTML<br>
  &lt;script src="https://packcdn.../cdn/abc123/index.js"&gt;&lt;/script>&lt;br><br>
  // or with curl<br>
  curl https://packcdn.../cdn/abc123/index.js
</div>
</section>
</div>

<div class="back-to-top" id="backToTop" onclick="scrollToTop()">
  ↑
</div>

<footer>
  <p>PackCDN © 2024 | Built with Cloudflare Workers, Vercel, and Supabase</p>
  <p style="margin-top: 10px;">
```

```
    <a href="selector.html">Switch to Classic View</a> |  
    <a href="Pages/Home/index.html">Go to Homepage</a>  
</p>  
</footer>
```

```
<script>  
  // Simple scroll animations  
  function initScrollAnimations() {  
    const sections = document.querySelectorAll('.section');  
  
    const observer = new IntersectionObserver((entries) => {  
      entries.forEach(entry => {  
        if (entry.isIntersecting) {  
          entry.target.classList.add('visible');  
        }  
      });  
    }, {  
      threshold: 0.1  
    });  
  
    sections.forEach(section => {  
      observer.observe(section);  
    });  
  }  
  
  // Flow diagram  
  function createFlowDiagram() {  
    const container = document.getElementById('flow-diagram');  
    const nodes = [  
      { id: 1, title: 'Editor', x: '10%', y: '20%', color: '#667eea' },  
      { id: 2, title: 'API', x: '30%', y: '20%', color: '#764ba2' },  
      { id: 3, title: 'Supabase', x: '50%', y: '60%', color: '#3ecf8e' },  
      { id: 4, title: 'Cloudflare', x: '70%', y: '20%', color: '#f6821f' },  
      { id: 5, title: 'User', x: '90%', y: '60%', color: '#1abc9c' }  
    ];  
  
    const connections = [  
      { from: 1, to: 2 },  
      { from: 2, to: 3 },  
      { from: 3, to: 4 },  
      { from: 4, to: 5 }  
    ];  
  
    // Create nodes
```

```

nodes.forEach(node => {
  const div = document.createElement('div');
  div.className = 'flow-node';
  div.textContent = node.title;
  div.style.left = node.x;
  div.style.top = node.y;
  div.style.borderColor = node.color;
  div.style.animationDelay = `${node.id * 0.5}s`;
  container.appendChild(div);
});
}

// Demo functions
let demoAnimation;

function runDemo() {
  const cube = document.getElementById('demoCube');
  if (demoAnimation) clearInterval(demoAnimation);

  let scale = 1;
  let growing = true;

  demoAnimation = setInterval(() => {
    if (growing) {
      scale += 0.05;
      if (scale >= 1.5) growing = false;
    } else {
      scale -= 0.05;
      if (scale <= 1) {
        clearInterval(demoAnimation);
        cube.style.transform = `scale(1)`;
        cube.style.background = 'linear-gradient(45deg, #667eea, #764ba2)';
        return;
      }
    }
  }, 50);

  cube.style.transform = `scale(${scale})`;
  cube.style.background = scale > 1.2
    ? 'linear-gradient(45deg, #ff6b6b, #ffd93d)'
    : 'linear-gradient(45deg, #667eea, #764ba2)';
}

function resetDemo() {

```

```

const cube = document.getElementById('demoCube');
if (demoAnimation) clearInterval(demoAnimation);

cube.style.transform = 'scale(1)';
cube.style.background = 'linear-gradient(45deg, #667eea, #764ba2)';
}

// Back to top button
function initBackToTop() {
  const backToTop = document.getElementById('backToTop');

  window.addEventListener('scroll', () => {
    if (window.scrollY > 500) {
      backToTop.style.opacity = '1';
      backToTop.style.transform = 'translateY(0)';
    } else {
      backToTop.style.opacity = '0';
      backToTop.style.transform = 'translateY(20px)';
    }
  });
}

function scrollToTop() {
  window.scrollTo({
    top: 0,
    behavior: 'smooth'
  });
}

// Smooth scrolling for anchor links
document.querySelectorAll('a[href^="#"]').forEach(anchor => {
  anchor.addEventListener('click', function (e) {
    e.preventDefault();
    const targetId = this.getAttribute('href');
    if (targetId === '#') return;

    const targetElement = document.querySelector(targetId);
    if (targetElement) {
      targetElement.scrollIntoView({
        behavior: 'smooth'
      });
    }
  });
});

```

```
// Initialize everything
window.addEventListener('DOMContentLoaded', () => {
  initScrollAnimations();
  createFlowDiagram();
  initBackToTop();

  // Make hero section visible
  document.querySelector('.hero-section').style.opacity = '1';

  console.log("Page loaded - NO libraries, pure CSS/JS!");
});
</script>
</body>
</html>

public/Docs/Pages/Home/index.html
<!DOCTYPE html>
<html>
<head>
  <title>PackCDN - Instant Package Publishing</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    * { margin: 0; padding: 0; box-sizing: border-box; }
    body { font-family: -apple-system, BlinkMacSystemFont, sans-serif; line-height: 1.6; color:
#333; }

    .hero {
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      color: white;
      text-align: center;
      padding: 80px 20px;
      clip-path: polygon(0 0, 100% 0, 100% 85%, 0 100%);
    }

    .hero h1 {
      font-size: 3.5rem;
      margin-bottom: 20px;
      font-weight: 800;
    }

    .hero p {
      font-size: 1.2rem;
      max-width: 600px;
```

```
margin: 0 auto 40px;  
opacity: 0.9;  
}
```

```
.cta-buttons {  
  display: flex;  
  gap: 15px;  
  justify-content: center;  
  flex-wrap: wrap;  
}
```

```
.btn {  
  padding: 15px 30px;  
  border-radius: 50px;  
  text-decoration: none;  
  font-weight: 600;  
  font-size: 1rem;  
  transition: all 0.3s ease;  
}
```

```
.btn-primary {  
  background: white;  
  color: #667eea;  
}
```

```
.btn-secondary {  
  background: transparent;  
  color: white;  
  border: 2px solid white;  
}
```

```
.btn:hover {  
  transform: translateY(-3px);  
  box-shadow: 0 10px 20px rgba(0,0,0,0.2);  
}
```

```
.features {  
  max-width: 1200px;  
  margin: -50px auto 100px;  
  padding: 0 20px;  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
  gap: 30px;  
}
```

```
.feature-card {  
  background: white;  
  padding: 40px 30px;  
  border-radius: 15px;  
  box-shadow: 0 10px 40px rgba(0,0,0,0.1);  
  text-align: center;  
}
```

```
.feature-icon {  
  font-size: 3rem;  
  margin-bottom: 20px;  
}
```

```
.feature-card h3 {  
  margin-bottom: 15px;  
  color: #2d3748;  
}
```

```
.how-it-works {  
  background: #f8f9fa;  
  padding: 100px 20px;  
  text-align: center;  
}
```

```
.steps {  
  max-width: 800px;  
  margin: 60px auto 0;  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: center;  
  gap: 40px;  
}
```

```
.step {  
  flex: 1;  
  min-width: 200px;  
  text-align: center;  
}
```

```
.step-number {  
  width: 60px;  
  height: 60px;  
  background: #667eea;
```

```
    color: white;
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 1.5rem;
    font-weight: bold;
    margin: 0 auto 20px;
}

.terminal {
    background: #1a1a1a;
    color: #00ff9d;
    padding: 20px;
    border-radius: 10px;
    font-family: monospace;
    max-width: 800px;
    margin: 40px auto;
    text-align: left;
    overflow-x: auto;
}

.terminal .comment { color: #666; }
.terminal .command { color: white; }

footer {
    background: #2d3748;
    color: white;
    text-align: center;
    padding: 40px 20px;
    margin-top: 100px;
}

@media (max-width: 768px) {
    .hero h1 { font-size: 2.5rem; }
    .hero { padding: 60px 20px; }
    .features { margin-top: -30px; }
}
</style>
</head>
<body>
<section class="hero">
  <h1>🚀 PackCDN</h1>
```

<p>Instantly publish and share custom packages. No registration required. Anonymous and free.</p>

<div class="cta-buttons">

Create a Pack

Explore Packs

</div>

</section>

<div class="features">

<div class="feature-card">

<div class="feature-icon"></div>

<h3>NPM Packages</h3>

<p>Publish JavaScript/TypeScript packages that can be imported directly in browsers or Node.js</p>

</div>

<div class="feature-card">

<div class="feature-icon"></div>

<h3>Python Modules</h3>

<p>Share Python code as importable modules with automatic dependency analysis</p>

</div>

<div class="feature-card">

<div class="feature-icon"></div>

<h3>WASM Modules</h3>

<p>Upload and serve WebAssembly modules for high-performance applications</p>

</div>

</div>

<section class="how-it-works">

<h2 style="font-size: 2.5rem; margin-bottom: 20px;">How It Works</h2>

<p style="max-width: 600px; margin: 0 auto 40px; color: #666;">

Three simple steps to publish and share your code

</p>

<div class="steps">

<div class="step">

<div class="step-number">1</div>

<h3>Write Code</h3>

<p>Use our built-in editor or upload existing files</p>

</div>

<div class="step">

<div class="step-number">2</div>

```
<h3>Publish</h3>
<p>Get instant CDN URLs and installation commands</p>
</div>
```

```
<div class="step">
  <div class="step-number">3</div>
  <h3>Share & Use</h3>
  <p>Share your pack URL or install directly</p>
</div>
</div>
</section>
```

```
<div style="max-width: 1200px; margin: 0 auto; padding: 0 20px;">
  <div class="terminal">
    <span class="comment"># Create and publish a package</span><br>
    <span class="command">$ pack install my-utils
https://packcdn.firefly-worker.workers.dev/cdn/abc123</span><br><br>

    <span class="comment"># Use in JavaScript</span><br>
    <span class="command">import utils from
'https://packcdn.firefly-worker.workers.dev/cdn/abc123/index.js';</span><br><br>

    <span class="comment"># Use in HTML</span><br>
    <span class="command">&lt;script
src="https://packcdn.firefly-worker.workers.dev/cdn/abc123/index.js"&gt;&lt;/script></span>
  </div>
</div>
```

```
<footer>
  <p style="margin-bottom: 20px; font-size: 1.2rem;">PackCDN - Instant Package
Publishing</p>
  <p style="color: #a0aec0; max-width: 600px; margin: 0 auto;">
    Built with Vercel, Cloudflare Workers, and Supabase.<br>
    All users are anonymous. No registration required. Free forever.
  </p>
  <div style="margin-top: 30px;">
    <a href="Pages/Editor/editor.html" style="color: white; margin: 0 15px;">Create</a>
    <a href="Pages/Explore/explore.html" style="color: white; margin: 0 15px;">Explore</a>
    <a href="https://github.com/sussybocca/PackCDN/tree/main" style="color: white; margin: 0
15px;">GitHub</a>
  </div>
</footer>
</body>
</html>
```

public/Docs/Pages/Home/Pages/Editor/editor.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Pack Editor - WebAssembly Support</title>
  <link rel="stylesheet" data-name="vs/editor/editor.main"
href="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/editor/editor.main.min.c
ss">
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    body {
      background-color: #1e1e1e;
      color: #d4d4d4;
      display: flex;
      flex-direction: column;
      height: 100vh;
      overflow: hidden;
    }

    header {
      background-color: #252526;
      padding: 15px 20px;
      display: flex;
      justify-content: space-between;
      align-items: center;
      border-bottom: 1px solid #3e3e42;
    }

    .logo {
      font-size: 1.5rem;
      font-weight: bold;
      color: #569cd6;
    }
```

```
.controls {
  display: flex;
  gap: 10px;
}

button {
  background-color: #0e639c;
  color: white;
  border: none;
  padding: 8px 16px;
  border-radius: 4px;
  cursor: pointer;
  font-weight: 500;
  transition: background-color 0.2s;
}

button:hover {
  background-color: #1177bb;
}

button.secondary {
  background-color: #3e3e42;
}

button.secondary:hover {
  background-color: #4a4a4f;
}

button.success {
  background-color: #388a34;
}

button.success:hover {
  background-color: #3fa33a;
}

button.disabled {
  background-color: #3e3e42;
  cursor: not-allowed;
  opacity: 0.6;
}

.container {
  display: flex;
```

```
    flex: 1;
    overflow: hidden;
}

.sidebar {
    width: 280px;
    background-color: #252526;
    border-right: 1px solid #3e3e42;
    padding: 20px;
    overflow-y: auto;
}

.sidebar h3 {
    margin-bottom: 15px;
    color: #cccccc;
    font-size: 1rem;
    font-weight: 500;
}

.file-list {
    list-style-type: none;
    margin-bottom: 30px;
    max-height: 200px;
    overflow-y: auto;
}

.file-list li {
    padding: 8px 12px;
    border-radius: 4px;
    margin-bottom: 5px;
    cursor: pointer;
    display: flex;
    align-items: center;
    justify-content: space-between;
}

.file-list li:hover {
    background-color: #2a2d2e;
}

.file-list li.active {
    background-color: #094771;
}
```

```
.file-icon {
  color: #c586c0;
  font-size: 0.9rem;
}

.json-file .file-icon {
  color: #f5d76e;
}

.js-file .file-icon {
  color: #4ec9b0;
}

.wasm-file .file-icon {
  color: #d16969;
}

.file-actions {
  display: flex;
  gap: 5px;
  opacity: 0;
  transition: opacity 0.2s;
}

.file-list li:hover .file-actions {
  opacity: 1;
}

.file-action-btn {
  background: none;
  border: none;
  color: #858585;
  padding: 2px 5px;
  font-size: 0.8rem;
}

.file-action-btn:hover {
  color: #d4d4d4;
}

.wasm-section {
  margin-top: 20px;
  border-top: 1px solid #3e3e42;
  padding-top: 20px;
}
```

```
}

.wasm-info {
  background-color: #2a2d2e;
  padding: 10px;
  border-radius: 4px;
  margin-bottom: 15px;
  font-size: 0.85rem;
}

.wasm-info.warning {
  background-color: #5c2d0c;
  border-left: 3px solid #f5d76e;
}

.wasm-info.success {
  background-color: #2d5c0c;
  border-left: 3px solid #3fa33a;
}

.upload-area {
  border: 2px dashed #3e3e42;
  border-radius: 8px;
  padding: 20px;
  text-align: center;
  cursor: pointer;
  transition: border-color 0.2s;
  margin-bottom: 15px;
}

.upload-area:hover {
  border-color: #0e639c;
}

.upload-area.dragover {
  border-color: #569cd6;
  background-color: rgba(86, 156, 214, 0.1);
}

.upload-icon {
  font-size: 2rem;
  color: #569cd6;
  margin-bottom: 10px;
}
```

```
.upload-area p {  
  margin-bottom: 5px;  
}
```

```
.upload-hint {  
  font-size: 0.8rem;  
  color: #858585;  
}
```

```
.wasm-list {  
  list-style-type: none;  
  margin-top: 10px;  
  max-height: 150px;  
  overflow-y: auto;  
}
```

```
.wasm-list li {  
  padding: 8px 12px;  
  background-color: #2a2d2e;  
  border-radius: 4px;  
  margin-bottom: 5px;  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  font-size: 0.9rem;  
}
```

```
.wasm-size {  
  color: #858585;  
  font-size: 0.8rem;  
}
```

```
.editor-container {  
  flex: 1;  
  display: flex;  
  flex-direction: column;  
  overflow: hidden;  
}
```

```
.editor-tabs {  
  display: flex;  
  background-color: #2d2d30;  
  border-bottom: 1px solid #3e3e42;
```

```
        overflow-x: auto;
    }

    .tab {
        padding: 10px 20px;
        border-right: 1px solid #3e3e42;
        cursor: pointer;
        white-space: nowrap;
        display: flex;
        align-items: center;
        gap: 8px;
    }

    .tab:hover {
        background-color: #2a2d2e;
    }

    .tab.active {
        background-color: #1e1e1e;
    }

    .close-tab {
        margin-left: 5px;
        opacity: 0.7;
    }

    .close-tab:hover {
        opacity: 1;
    }

    #editor {
        flex: 1;
        overflow: hidden;
    }

    .config-panel {
        margin-top: 20px;
    }

    .config-item {
        margin-bottom: 15px;
    }

    label {
```

```
display: block;
margin-bottom: 5px;
font-size: 0.9rem;
color: #cccccc;
}

input[type="text"], select {
width: 100%;
padding: 8px 12px;
background-color: #3e3e42;
border: 1px solid #565656;
border-radius: 4px;
color: #d4d4d4;
}

input[type="text"]:focus, select:focus {
outline: none;
border-color: #007acc;
}

.checkbox-group {
display: flex;
align-items: center;
gap: 8px;
margin-top: 5px;
}

input[type="checkbox"] {
accent-color: #0e639c;
}

.publish-modal {
position: fixed;
top: 0;
left: 0;
width: 100%;
height: 100%;
background-color: rgba(0, 0, 0, 0.8);
display: flex;
justify-content: center;
align-items: center;
z-index: 1000;
display: none;
}
```

```
.modal-content {  
  background-color: #252526;  
  border-radius: 8px;  
  padding: 25px;  
  width: 90%;  
  max-width: 500px;  
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.5);  
}
```

```
.modal-header {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  margin-bottom: 20px;  
}
```

```
.modal-header h2 {  
  color: #569cd6;  
  font-size: 1.5rem;  
}
```

```
.close-modal {  
  font-size: 1.8rem;  
  cursor: pointer;  
  color: #cccccc;  
}
```

```
.close-modal:hover {  
  color: white;  
}
```

```
.publish-result {  
  background-color: #1e1e1e;  
  padding: 15px;  
  border-radius: 4px;  
  margin-top: 20px;  
  font-family: monospace;  
  font-size: 0.9rem;  
  white-space: pre-wrap;  
  word-break: break-all;  
  display: none;  
}
```

```
.result-item {  
    margin-bottom: 10px;  
}
```

```
.result-label {  
    color: #9cdcfe;  
    font-weight: bold;  
}
```

```
.result-value {  
    color: #ce9178;  
}
```

```
.result-value a {  
    color: #569cd6;  
    text-decoration: none;  
}
```

```
.result-value a:hover {  
    text-decoration: underline;  
}
```

```
.loading {  
    display: none;  
    text-align: center;  
    margin: 20px 0;  
    color: #569cd6;  
}
```

```
.spinner {  
    border: 3px solid rgba(86, 156, 214, 0.3);  
    border-radius: 50%;  
    border-top: 3px solid #569cd6;  
    width: 30px;  
    height: 30px;  
    animation: spin 1s linear infinite;  
    margin: 0 auto 10px;  
}
```

```
@keyframes spin {  
    0% { transform: rotate(0deg); }  
    100% { transform: rotate(360deg); }  
}
```

```
.notification {
  position: fixed;
  bottom: 20px;
  right: 20px;
  padding: 12px 20px;
  background-color: #0c7b93;
  color: white;
  border-radius: 4px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
  z-index: 1001;
  transform: translateY(100px);
  opacity: 0;
  transition: transform 0.3s, opacity 0.3s;
}

.notification.show {
  transform: translateY(0);
  opacity: 1;
}

.notification.error {
  background-color: #a12c2c;
}

.notification.success {
  background-color: #388a34;
}

footer {
  padding: 10px 20px;
  background-color: #252526;
  border-top: 1px solid #3e3e42;
  font-size: 0.8rem;
  color: #858585;
  text-align: center;
}
</style>
</head>
<body>
  <header>
    <div class="logo">Pack Editor</div>
    <div class="controls">
      <button id="newFileBtn" class="secondary">New File</button>
      <button id="saveBtn">Save</button>
    </div>
  </header>
</body>
</html>
```

```

        <button id="publishBtn" class="success">Publish Pack</button>
    </div>
</header>

<div class="container">
    <div class="sidebar">
        <h3>Files</h3>
        <ul class="file-list" id="fileList">
            <!-- Files will be added here dynamically -->
        </ul>

        <div class="wasm-section">
            <h3>WebAssembly Files</h3>

            <div class="wasm-info warning" id="wasmInfo">
                <strong>Important:</strong> WASM files cannot be edited directly. You can only
upload valid WebAssembly binary files (.wasm). These will be served from the CDN.
            </div>

            <div class="upload-area" id="uploadArea">
                <div class="upload-icon">⬆</div>
                <p>Drag & drop WASM files here</p>
                <p class="upload-hint">or click to browse</p>
                <p class="upload-hint">Only .wasm files accepted</p>
            </div>

            <ul class="wasm-list" id="wasmList">
                <!-- WASM files will be listed here -->
            </ul>
        </div>

        <div class="config-panel">
            <h3>Pack Configuration</h3>

            <div class="config-item">
                <label for="packName">Pack Name</label>
                <input type="text" id="packName" placeholder="my-awesome-pack"
value="my-awesome-pack">
            </div>

            <div class="config-item">
                <label for="packType">Package Type</label>
                <select id="packType">
                    <option value="module">Module</option>

```

```

        <option value="library">Library</option>
        <option value="template">Template</option>
        <option value="plugin">Plugin</option>
    </select>
</div>

<div class="config-item">
    <label>Visibility</label>
    <div class="checkbox-group">
        <input type="checkbox" id="isPublic" checked>
        <label for="isPublic">Public Package</label>
    </div>
</div>
</div>
</div>

<div class="editor-container">
    <div class="editor-tabs" id="editorTabs">
        <!-- Editor tabs will be added here dynamically -->
    </div>
    <div id="editor"></div>
</div>
</div>

<div class="publish-modal" id="publishModal">
    <div class="modal-content">
        <div class="modal-header">
            <h2>Publish Pack</h2>
            <div class="close-modal" id="closeModal">&times;</div>
        </div>

        <p>Ready to publish your pack? This will upload all files and generate a unique URL for
your package.</p>

        <div id="validationResults">
            <!-- Validation results will appear here -->
        </div>

        <div class="loading" id="publishLoading">
            <div class="spinner"></div>
            <p>Publishing your pack...</p>
        </div>

        <div class="publish-result" id="publishResult">

```

```

        <!-- Publish results will be displayed here -->
    </div>

    <div style="display: flex; gap: 10px; margin-top: 20px;">
        <button id="confirmPublishBtn" class="success" style="flex: 1;">Publish
Now</button>
        <button id="cancelPublishBtn" class="secondary">Cancel</button>
    </div>
</div>
</div>

<div class="notification" id="notification">Message</div>

<footer>
    Pack Editor - Edit and publish packages with WebAssembly support
</footer>

<!-- Monaco Editor loader -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/loader.min.js"></script>
<script>
    // Monaco editor configuration
    require.config({ paths: { vs:
'https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs' } });

    // Initialize app state
    const appState = {
        files: {
            'package.json': {
                name: 'package.json',
                content: JSON.stringify({
                    name: 'my-awesome-pack',
                    version: '1.0.0',
                    type: 'module',
                    description: 'My awesome package with WASM support',
                    main: 'index.js',
                    author: '',
                    license: 'MIT'
                }, null, 2),
                language: 'json',
                editable: true
            },
            'index.js': {
                name: 'index.js',

```

```
    content: `// Main entry point for your package

export function helloWorld() {
  console.log('Hello from my awesome pack!');
  return 'Hello World!';
}

// Add your code here
export const version = '1.0.0';

// Example utility function
export function sum(a, b) {
  return a + b;
}`,
  language: 'javascript',
  editable: true
},
'README.md': {
  name: 'README.md',
  content: `# My Awesome Pack
```

This is a sample package created with Pack Editor.

Installation

```
```bash
pack install my-awesome-pack
```
```

Usage

```
```javascript
import { helloWorld } from 'my-awesome-pack';

helloWorld(); // Outputs: Hello from my awesome pack!
```
```

API

helloWorld()

Prints a greeting message and returns "Hello World!"

sum(a, b)

Returns the sum of two numbers

License

MIT

,

```
    language: 'markdown',
    editable: true
  },
  wasmFiles: {},
  activeFile: 'package.json',
  editor: null,
  packName: 'my-awesome-pack',
  packType: 'module',
  isPublic: true
};
```

// DOM Elements

```
const fileList = document.getElementById('fileList');
const editorTabs = document.getElementById('editorTabs');
const editorContainer = document.getElementById('editor');
const packNameInput = document.getElementById('packName');
const packTypeSelect = document.getElementById('packType');
const isPublicCheckbox = document.getElementById('isPublic');
const publishModal = document.getElementById('publishModal');
const closeModal = document.getElementById('closeModal');
const cancelPublishBtn = document.getElementById('cancelPublishBtn');
const confirmPublishBtn = document.getElementById('confirmPublishBtn');
const publishResult = document.getElementById('publishResult');
const publishLoading = document.getElementById('publishLoading');
const notification = document.getElementById('notification');
const uploadArea = document.getElementById('uploadArea');
const wasmList = document.getElementById('wasmList');
const wasmInfo = document.getElementById('wasmInfo');
const validationResults = document.getElementById('validationResults');
```

// File input for WASM uploads (hidden)

```
const fileInput = document.createElement('input');
fileInput.type = 'file';
fileInput.accept = '.wasm';
fileInput.multiple = true;
fileInput.style.display = 'none';
document.body.appendChild(fileInput);
```

```

// Initialize Monaco Editor
require(['vs/editor/editor.main'], function() {
  // Create editor instance
  appState.editor = monaco.editor.create(editorContainer, {
    value: appState.files['package.json'].content,
    language: 'json',
    theme: 'vs-dark',
    minimap: { enabled: true },
    scrollBeyondLastLine: false,
    automaticLayout: true,
    fontSize: 14,
    lineNumbers: 'on',
    roundedSelection: false,
    scrollbar: {
      vertical: 'auto',
      horizontal: 'auto'
    },
    readOnly: false
  });

  // Update app state when editor content changes
  appState.editor.onDidChangeModelContent(() => {
    const activeFile = appState.activeFile;
    if (activeFile && appState.files[activeFile]) {
      appState.files[activeFile].content = appState.editor.getValue();

      // If this is package.json, update the pack name if changed
      if (activeFile === 'package.json') {
        try {
          const packageJson = JSON.parse(appState.editor.getValue());
          if (packageJson.name && packageJson.name !== appState.packName) {
            appState.packName = packageJson.name;
            packNameInput.value = packageJson.name;
          }
          if (packageJson.type && packageJson.type !== appState.packType) {
            appState.packType = packageJson.type;
            packTypeSelect.value = packageJson.type;
          }
        } catch (e) {
          // Invalid JSON, ignore
        }
      }
    }
  });
});

```

```

// Initialize UI
renderFileList();
renderEditorTabs();
setupEventListeners();
});

// Render the file list in the sidebar
function renderFileList() {
  fileList.innerHTML = "";

  Object.keys(appState.files).forEach(fileName => {
    const file = appState.files[fileName];
    const li = document.createElement('li');
    li.className = `file-item ${fileName === appState.activeFile ? 'active' : ''}
${file.language}-file`;

    const fileExt = fileName.split('.').pop();
    const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

    li.innerHTML = `
      <div>
        <span class="file-icon">${icon}</span>
        <span>${fileName}</span>
      </div>
      <div class="file-actions">
        <button class="file-action-btn" data-file="${fileName}"
title="Rename">✏️</button>
        ${fileName !== 'package.json' && fileName !== 'index.js' ?
`<button class="file-action-btn" data-file="${fileName}"
title="Delete">🗑️</button>` : ''}
      </div>
    `;

    li.addEventListener('click', (e) => {
      if (!e.target.closest('.file-action-btn')) {
        switchFile(fileName);
      }
    });

    fileList.appendChild(li);
  });

  // Add event listeners for file actions

```

```

document.querySelectorAll('.file-action-btn').forEach(btn => {
  btn.addEventListener('click', (e) => {
    e.stopPropagation();
    const fileName = btn.dataset.file;

    if (btn.textContent.includes('✎')) {
      renameFile(fileName);
    } else if (btn.textContent.includes('🗑')) {
      deleteFile(fileName);
    }
  });
});
}

// Render WASM file list
function renderWasmList() {
  wasmList.innerHTML = "";

  if (Object.keys(appState.wasmFiles).length === 0) {
    const li = document.createElement('li');
    li.textContent = 'No WASM files uploaded';
    li.style.color = '#858585';
    li.style.fontStyle = 'italic';
    wasmList.appendChild(li);
    wasmInfo.className = 'wasm-info warning';
    wasmInfo.innerHTML = '<strong>Important:</strong> WASM files cannot be edited
directly. You can only upload valid WebAssembly binary files (.wasm). These will be served from
the CDN.';
  } else {
    Object.keys(appState.wasmFiles).forEach(fileName => {
      const file = appState.wasmFiles[fileName];
      const li = document.createElement('li');

      li.innerHTML = `
        <div>
          <span class="file-icon">⚡ </span>
          <span>${fileName}</span>
        </div>
        <div>
          <span class="wasm-size">${formatFileSize(file.size)}</span>
          <button class="file-action-btn" data-wasm="${fileName}"
title="Remove">🗑 </button>
        </div>
      `;
    });
  }
}

```

```

        wasmList.appendChild(li);
    });

    wasmInfo.className = 'wasm-info success';
    wasmInfo.innerHTML = `<strong>✓ Ready:</strong>
    ${Object.keys(appState.wasmFiles).length} WASM file(s) uploaded and validated.`;

    // Add event listeners for remove buttons
    document.querySelectorAll('[data-wasm]').forEach(btn => {
        btn.addEventListener('click', (e) => {
            const fileName = btn.dataset.wasm;
            deleteWasmFile(fileName);
        });
    });
}

// Render editor tabs
function renderEditorTabs() {
    editorTabs.innerHTML = "";

    Object.keys(appState.files).forEach(fileName => {
        const tab = document.createElement('div');
        tab.className = `tab ${fileName === appState.activeFile ? 'active' : ''}`;
        tab.dataset.file = fileName;

        const fileExt = fileName.split('.').pop();
        const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

        tab.innerHTML = `
            <span class="file-icon">${icon}</span>
            <span>${fileName}</span>
            ${fileName !== 'package.json' && fileName !== 'index.js' ?
                '<span class="close-tab" data-file="' + fileName + '">✕</span>' : ''}
        `;

        tab.addEventListener('click', (e) => {
            if (!e.target.classList.contains('close-tab')) {
                switchFile(fileName);
            }
        });

        editorTabs.appendChild(tab);
    });
}

```

```

});

// Add event listeners for close tab buttons
document.querySelectorAll('.close-tab').forEach(btn => {
  btn.addEventListener('click', (e) => {
    e.stopPropagation();
    const fileName = btn.dataset.file;
    deleteFile(fileName);
  });
});
}

// Switch to a different file
function switchFile(fileName) {
  if (!appState.files[fileName]) return;

  appState.activeFile = fileName;
  const file = appState.files[fileName];

  // Update editor content and language
  appState.editor.setValue(file.content);
  monaco.editor.setModelLanguage(appState.editor.getModel(), file.language);

  // Update UI
  renderFileList();
  renderEditorTabs();
}

// Create a new file
function createNewFile() {
  const fileName = prompt('Enter file name (with extension, e.g., utils.js):', 'newfile.js');
  if (!fileName) return;

  if (appState.files[fileName] || appState.wasmFiles[fileName]) {
    showNotification('File already exists!', 'error');
    return;
  }

  const fileExt = fileName.split('.').pop();
  let language = 'plaintext';
  let content = "";

  switch (fileExt) {
    case 'js':

```

```

        language = 'javascript';
        content = `// ${fileName}\n\n`;
        break;
    case 'json':
        language = 'json';
        content = `{ \n  \n }\n`;
        break;
    case 'md':
        language = 'markdown';
        content = `# ${fileName}\n\n`;
        break;
    case 'html':
        language = 'html';
        content = `<!DOCTYPE html>\n<html>\n<head>\n
<title></title>\n</head>\n<body>\n  \n</body>\n</html>\n`;
        break;
    case 'css':
        language = 'css';
        content = `/* ${fileName} */\n\n`;
        break;
    }

    appState.files[fileName] = {
        name: fileName,
        content,
        language,
        editable: true
    };

    switchFile(fileName);
    showNotification(`Created ${fileName}`, 'success');
}

// Rename a file
function renameFile(oldName) {
    const newName = prompt('Enter new file name:', oldName);
    if (!newName || newName === oldName) return;

    if (appState.files[newName] || appState.wasmFiles[newName]) {
        showNotification('File already exists!', 'error');
        return;
    }

    appState.files[newName] = {

```

```

        ...appState.files[oldName],
        name: newName
    };

    delete appState.files[oldName];

    if (appState.activeFile === oldName) {
        appState.activeFile = newName;
    }

    renderFileList();
    renderEditorTabs();
    showNotification(`Renamed ${oldName} to ${newName}`, 'success');
}

// Delete a file
function deleteFile(fileName) {
    if (fileName === 'package.json' || fileName === 'index.js') {
        showNotification('Cannot delete required files!', 'error');
        return;
    }

    if (confirm(`Are you sure you want to delete ${fileName}?`)) {
        delete appState.files[fileName];

        if (appState.activeFile === fileName) {
            const remainingFiles = Object.keys(appState.files);
            if (remainingFiles.length > 0) {
                switchFile(remainingFiles[0]);
            }
        }

        renderFileList();
        renderEditorTabs();
        showNotification(`Deleted ${fileName}`, 'success');
    }
}

// Handle WASM file upload
async function handleWasmUpload(files) {
    for (const file of files) {
        if (!file.name.endsWith('.wasm')) {
            showNotification(`Skipped ${file.name}: Not a .wasm file`, 'error');
            continue;
        }
    }
}

```

```

    }

    // Validate WASM file
    const isValid = await validateWasmFile(file);
    if (!isValid) {
        showNotification(`Skipped ${file.name}: Invalid WebAssembly file`, 'error');
        continue;
    }

    // Read file as ArrayBuffer
    const reader = new FileReader();
    reader.onload = async (e) => {
        const arrayBuffer = e.target.result;

        // Store WASM file
        appState.wasmFiles[file.name] = {
            name: file.name,
            size: file.size,
            data: arrayBuffer
        };

        renderWasmList();
        showNotification(`Uploaded ${file.name} (${formatFileSize(file.size)})`, 'success');
    };

    reader.readAsArrayBuffer(file);
}
}

// Validate a WASM file
async function validateWasmFile(file) {
    try {
        // Basic WASM validation - check magic number
        const arrayBuffer = await file.slice(0, 8).arrayBuffer();
        const header = new Uint8Array(arrayBuffer);

        // WASM magic number: \0asm
        const wasmMagic = [0x00, 0x61, 0x73, 0x6D];

        for (let i = 0; i < 4; i++) {
            if (header[i] !== wasmMagic[i]) {
                return false;
            }
        }
    }
}

```

```

        return true;
    } catch (error) {
        console.error('WASM validation error:', error);
        return false;
    }
}

// Delete a WASM file
function deleteWasmFile(fileName) {
    if (confirm(`Are you sure you want to remove ${fileName}?`)) {
        delete appState.wasmFiles[fileName];
        renderWasmList();
        showNotification(`Removed ${fileName}`, 'success');
    }
}

// Format file size
function formatFileSize(bytes) {
    if (bytes === 0) return '0 Bytes';
    const k = 1024;
    const sizes = ['Bytes', 'KB', 'MB', 'GB'];
    const i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i];
}

// Show notification
function showNotification(message, type = 'info') {
    notification.textContent = message;
    notification.className = `notification ${type}`;
    notification.classList.add('show');

    setTimeout(() => {
        notification.classList.remove('show');
    }, 3000);
}

// Validate package before publishing
function validatePackage() {
    const errors = [];
    const warnings = [];

    // Check package.json
    try {

```

```

const packageJson = JSON.parse(appState.files['package.json'].content);

if (!packageJson.name || packageJson.name.trim() === '') {
  errors.push('Package name is required in package.json');
}

if (!packageJson.version) {
  warnings.push('Consider adding a version field to package.json');
}

if (!packageJson.main) {
  warnings.push('Consider specifying a main entry point in package.json');
}
} catch (e) {
  errors.push('Invalid JSON in package.json');
}

// Check if index.js exists
if (!appState.files['index.js']) {
  warnings.push('index.js is recommended as the main entry point');
}

// Check for invalid file names
Object.keys(appState.files).forEach(fileName => {
  if (fileName.includes(' ')) {
    warnings.push(`File "${fileName}" contains spaces, consider using hyphens or
underscores`);
  }
});

return { errors, warnings };
}

// Publish package to API - FIXED FOR API EXPECTATIONS
async function publishPackage() {
  // Validate package first
  const validation = validatePackage();

  if (validation.errors.length > 0) {
    showNotification('Cannot publish: ' + validation.errors[0], 'error');
    return;
  }

  // Show loading state

```

```

publishLoading.style.display = 'block';
validationResults.innerHTML = "";

try {
  // Prepare package.json
  const packageJson = JSON.parse(appState.files['package.json'].content);
  packageJson.name = appState.packName;
  packageJson.type = appState.packType;

  // Prepare all files for upload - SIMPLE OBJECT AS API EXPECTS
  const files = {};

  // Add regular files - just the content as string
  Object.keys(appState.files).forEach(fileName => {
    files[fileName] = appState.files[fileName].content;
  });

  // Add WASM files - convert ArrayBuffer to base64 string
  Object.keys(appState.wasmFiles).forEach(fileName => {
    const wasmFile = appState.wasmFiles[fileName];
    // Convert ArrayBuffer to base64 string
    const bytes = new Uint8Array(wasmFile.data);
    let binary = "";
    for (let i = 0; i < bytes.byteLength; i++) {
      binary += String.fromCharCode(bytes[i]);
    }
    files[fileName] = btoa(binary); // Just the base64 string, not an object
  });

  // Prepare request body - EXACTLY AS API EXPECTS
  const requestBody = {
    name: appState.packName,
    packJson: JSON.stringify(packageJson), // Already a string in your API
    files: files, // Simple object: {filename: contentString}
    isPublic: appState.isPublic
  };

  console.log('Sending to API:', {
    name: requestBody.name,
    packJson: requestBody.packJson.substring(0, 100) + '...',
    fileCount: Object.keys(requestBody.files).length,
    fileNames: Object.keys(requestBody.files)
  });
}

```

```

// Call publish API
const response = await fetch('/api/publish.js', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(requestBody)
});

const responseText = await response.text();
console.log('API Response:', responseText);

if (!response.ok) {
  throw new Error(`HTTP ${response.status}: ${response.statusText}. Response:
${responseText}`);
}

const result = JSON.parse(responseText);

// Hide loading, show results
publishLoading.style.display = 'none';
publishResult.style.display = 'block';

// Display results
publishResult.innerHTML = `
  <div class="result-item">
    <span class="result-label">✓ Published successfully!</span>
  </div>
  <div class="result-item">
    <span class="result-label">Pack ID:</span>
    <span class="result-value">${result.packId}</span>
  </div>
  <div class="result-item">
    <span class="result-label">CDN URL:</span>
    <span class="result-value"><a href="${result.cdnUrl}"
target="_blank">${result.cdnUrl}</a></span>
  </div>
  <div class="result-item">
    <span class="result-label">Worker URL:</span>
    <span class="result-value"><a href="${result.workerUrl}"
target="_blank">${result.workerUrl}</a></span>
  </div>
  <div class="result-item">
    <span class="result-label">Install Command:</span>

```

```

        <span class="result-value">${result.installCommand}</span>
    </div>
    ${result.encryptedKey ? `
    <div class="result-item">
        <span class="result-label">Encryption Key:</span>
        <span class="result-value">${result.encryptedKey}</span>
    </div>
    <div class="result-item">
        <span class="result-label">⚠ Save this key! It won't be shown again.</span>
    </div>
    ` : ""}
    <div class="result-item">
        <button id="exploreBtn" class="success" style="width: 100%; margin-top:
10px;">Explore Package</button>
    </div>
    `;

    // Add event listener for explore button
    document.getElementById('exploreBtn').addEventListener('click', () => {
        window.location.href = 'explore.html';
    });

    } catch (error) {
        publishLoading.style.display = 'none';
        showNotification(`Publish failed: ${error.message}`, 'error');
        console.error('Publish error:', error);
    }
}

// Setup event listeners
function setupEventListeners() {
    // Pack configuration
    packNameInput.addEventListener('input', (e) => {
        appState.packName = e.target.value;
    });

    packTypeSelect.addEventListener('change', (e) => {
        appState.packType = e.target.value;
    });

    isPublicCheckbox.addEventListener('change', (e) => {
        appState.isPublic = e.target.checked;
    });
}

```

```

// New file button
document.getElementById('newFileBtn').addEventListener('click', createNewFile);

// Save button
document.getElementById('saveBtn').addEventListener('click', () => {
  showNotification('All changes saved locally', 'success');
});

// Publish button
document.getElementById('publishBtn').addEventListener('click', () => {
  // Validate before showing modal
  const validation = validatePackage();

  validationResults.innerHTML = "";

  if (validation.errors.length > 0) {
    validationResults.innerHTML = `
      <div style="background-color: #5c2d0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
        <strong>Errors:</strong>
        <ul style="margin: 5px 0 0 20px;">
          ${validation.errors.map(err => `<li>${err}</li>`).join("")}
        </ul>
      </div>
    `;
  } else if (validation.warnings.length > 0) {
    validationResults.innerHTML = `
      <div style="background-color: #5c5c0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
        <strong>Warnings:</strong>
        <ul style="margin: 5px 0 0 20px;">
          ${validation.warnings.map(warn => `<li>${warn}</li>`).join("")}
        </ul>
      </div>
    `;
  }

  publishModal.style.display = 'flex';
  publishResult.style.display = 'none';
  publishLoading.style.display = 'none';
});

// Modal controls
closeModal.addEventListener('click', () => {

```

```

    publishModal.style.display = 'none';
  });

  cancelPublishBtn.addEventListener('click', () => {
    publishModal.style.display = 'none';
  });

  confirmPublishBtn.addEventListener('click', publishPackage);

  // WASM file upload
  uploadArea.addEventListener('click', () => {
    fileInput.click();
  });

  fileInput.addEventListener('change', (e) => {
    handleWasmUpload(Array.from(e.target.files));
    fileInput.value = "";
  });

  // Drag and drop for WASM files
  uploadArea.addEventListener('dragover', (e) => {
    e.preventDefault();
    uploadArea.classList.add('dragover');
  });

  uploadArea.addEventListener('dragleave', () => {
    uploadArea.classList.remove('dragover');
  });

  uploadArea.addEventListener('drop', (e) => {
    e.preventDefault();
    uploadArea.classList.remove('dragover');

    const files = Array.from(e.dataTransfer.files);
    handleWasmUpload(files);
  });
}

// Initialize WASM list
renderWasmList();

// Close modal when clicking outside
window.addEventListener('click', (e) => {
  if (e.target === publishModal) {

```

```
        publishModal.style.display = 'none';
    }
    });
</script>
</body>
</html>
```

public/Docs/Pages/Home/Pages/Explore/explore.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Explore Packs - PackCDN</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body { margin: 0; font-family: -apple-system, BlinkMacSystemFont, sans-serif; background:
    #f8f9fa; }
    .container { max-width: 1200px; margin: 0 auto; padding: 20px; }
    header { background: white; padding: 20px; border-radius: 10px; margin-bottom: 30px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1); }
    .search-box { display: flex; gap: 10px; margin-top: 20px; }
    .search-box input { flex: 1; padding: 12px; border: 1px solid #ddd; border-radius: 5px;
    font-size: 16px; }
    .search-box button { padding: 12px 30px; background: #667eea; color: white; border: none;
    border-radius: 5px; cursor: pointer; }
    .filters { display: flex; gap: 10px; margin-top: 15px; }
    .filter-btn { padding: 8px 16px; background: white; border: 1px solid #ddd; border-radius:
    20px; cursor: pointer; }
    .filter-btn.active { background: #667eea; color: white; border-color: #667eea; }
    .packs-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(300px, 1fr)); gap:
    20px; margin-top: 30px; }
    .pack-card { background: white; border-radius: 10px; padding: 20px; box-shadow: 0 2px 10px
    rgba(0,0,0,0.1); cursor: pointer; transition: transform 0.2s; }
    .pack-card:hover { transform: translateY(-5px); }
    .pack-header { display: flex; justify-content: space-between; align-items: start; }
    .pack-type { background: #e9ecef; color: #495057; padding: 4px 10px; border-radius: 15px;
    font-size: 12px; }
    .pack-name { font-size: 18px; font-weight: bold; margin: 10px 0; }
    .pack-meta { color: #6c757d; font-size: 14px; }
    .pack-stats { display: flex; gap: 15px; margin-top: 15px; font-size: 14px; }
    .pack-stat { display: flex; align-items: center; gap: 5px; }
    .empty-state { text-align: center; padding: 60px 20px; color: #6c757d; }
    .load-more { text-align: center; margin: 40px 0; }
```

```

    .load-btn { padding: 12px 40px; background: white; border: 2px solid #667eea; color:
#667eea; border-radius: 5px; cursor: pointer; font-size: 16px; }
</style>
</head>
<body>
  <div class="container">
    <header>
      <h1>Explore Packs</h1>
      <p>Discover and use packages created by the community</p>

      <div class="search-box">
        <input type="text" id="searchInput" placeholder="Search packages...">
        <button onclick="searchPacks()">Search</button>
      </div>

      <div class="filters">
        <div class="filter-btn active" data-type="all" onclick="setFilter('all')">All</div>
        <div class="filter-btn" data-type="npm" onclick="setFilter('npm')">NPM</div>
        <div class="filter-btn" data-type="python" onclick="setFilter('python')">Python</div>
        <div class="filter-btn" data-type="wasm" onclick="setFilter('wasm')">WASM</div>
      </div>
    </header>

    <div id="packsContainer" class="packs-grid">
      <!-- Packs will be loaded here -->
    </div>

    <div id="emptyState" class="empty-state" style="display: none;">
      <h3>No packs found</h3>
      <p>Try a different search or create your own pack!</p>
      <a href="editor.html" style="color: #667eea; text-decoration: none;">Create a Pack →</a>
    </div>

    <div class="load-more">
      <button class="load-btn" onclick="loadMore()" id="loadMoreBtn" style="display:
none;">Load More</button>
    </div>

    <script>
      let currentPage = 1;
      let currentFilter = 'all';
      let currentSearch = '';
      let hasMore = false;

```

```

// Load packs on page load
window.onload = loadPacks;

// Enter key to search
document.getElementById('searchInput').addEventListener('keypress', (e) => {
  if (e.key === 'Enter') searchPacks();
});

async function loadPacks(reset = false) {
  if (reset) {
    currentPage = 1;
    document.getElementById('packsContainer').innerHTML = "";
    document.getElementById('loadMoreBtn').style.display = 'none';
  }

  const params = new URLSearchParams({
    page: currentPage,
    limit: 12
  });

  if (currentFilter !== 'all') params.set('type', currentFilter);
  if (currentSearch) params.set('q', currentSearch);

  const response = await fetch(`/api/search?${params}`);
  const result = await response.json();

  if (result.success) {
    displayPacks(result.packs);
    hasMore = result.hasMore;

    if (hasMore) {
      document.getElementById('loadMoreBtn').style.display = 'block';
    }

    // Show/hide empty state
    if (result.packs.length === 0 && currentPage === 1) {
      document.getElementById('emptyState').style.display = 'block';
    } else {
      document.getElementById('emptyState').style.display = 'none';
    }
  }
}

```

```

function displayPacks(packs) {
  const container = document.getElementById('packsContainer');

  packs.forEach(pack => {
    const card = document.createElement('div');
    card.className = 'pack-card';
    card.onclick = () =>
      window.open(`https://packcdn.firefly-worker.workers.dev/pack/${pack.id}`, '_blank');

    card.innerHTML = `
      <div class="pack-header">
        <span class="pack-type">${pack.package_type.toUpperCase()}</span>
        <span class="pack-meta">v${pack.version}</span>
      </div>
      <div class="pack-name">${pack.name}</div>
      <div class="pack-stats">
        <div class="pack-stat">👁️ ${pack.views || 0} views</div>
        <div class="pack-stat">📦 ${pack.downloads || 0} downloads</div>
      </div>
      <div style="margin-top: 15px; font-size: 12px; color: #999;">
        Created ${formatDate(pack.created_at)}
      </div>
    `;

    container.appendChild(card);
  });
}

function searchPacks() {
  currentSearch = document.getElementById('searchInput').value;
  loadPacks(true);
}

function setFilter(type) {
  currentFilter = type;
  document.querySelectorAll('.filter-btn').forEach(btn => {
    btn.classList.toggle('active', btn.dataset.type === type);
  });
  loadPacks(true);
}

function loadMore() {
  currentPage++;
  loadPacks(false);
}

```

```

    }

    function formatDate(dateString) {
        const date = new Date(dateString);
        const now = new Date();
        const diff = now - date;

        const minutes = Math.floor(diff / 60000);
        const hours = Math.floor(diff / 3600000);
        const days = Math.floor(diff / 86400000);

        if (minutes < 60) return `${minutes}m ago`;
        if (hours < 24) return `${hours}h ago`;
        if (days < 7) return `${days}d ago`;

        return date.toLocaleDateString();
    }
</script>
</body>
</html>

public/Editor/editor.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pack Editor - WebAssembly Support</title>
    <link rel="stylesheet" data-name="vs/editor/editor.main"
href="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/editor/editor.main.min.c
ss">
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #1e1e1e;
            color: #d4d4d4;
            display: flex;
            flex-direction: column;

```

```
    height: 100vh;
    overflow: hidden;
}

header {
    background-color: #252526;
    padding: 15px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px solid #3e3e42;
}

.logo {
    font-size: 1.5rem;
    font-weight: bold;
    color: #569cd6;
}

.controls {
    display: flex;
    gap: 10px;
}

button {
    background-color: #0e639c;
    color: white;
    border: none;
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    font-weight: 500;
    transition: background-color 0.2s;
}

button:hover {
    background-color: #1177bb;
}

button.secondary {
    background-color: #3e3e42;
}

button.secondary:hover {
```

```
        background-color: #4a4a4f;
    }

    button.success {
        background-color: #388a34;
    }

    button.success:hover {
        background-color: #3fa33a;
    }

    button.disabled {
        background-color: #3e3e42;
        cursor: not-allowed;
        opacity: 0.6;
    }

    .container {
        display: flex;
        flex: 1;
        overflow: hidden;
    }

    .sidebar {
        width: 320px;
        background-color: #252526;
        border-right: 1px solid #3e3e42;
        padding: 20px;
        overflow-y: auto;
    }

    .sidebar h3 {
        margin-bottom: 15px;
        color: #cccccc;
        font-size: 1rem;
        font-weight: 500;
    }

    .file-list {
        list-style-type: none;
        margin-bottom: 30px;
        max-height: 200px;
        overflow-y: auto;
    }
```

```
.file-list li {  
  padding: 8px 12px;  
  border-radius: 4px;  
  margin-bottom: 5px;  
  cursor: pointer;  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
}
```

```
.file-list li:hover {  
  background-color: #2a2d2e;  
}
```

```
.file-list li.active {  
  background-color: #094771;  
}
```

```
.file-icon {  
  color: #c586c0;  
  font-size: 0.9rem;  
}
```

```
.json-file .file-icon {  
  color: #f5d76e;  
}
```

```
.js-file .file-icon {  
  color: #4ec9b0;  
}
```

```
.wasm-file .file-icon {  
  color: #d16969;  
}
```

```
.file-actions {  
  display: flex;  
  gap: 5px;  
  opacity: 0;  
  transition: opacity 0.2s;  
}
```

```
.file-list li:hover .file-actions {
```

```
    opacity: 1;
}

.file-action-btn {
    background: none;
    border: none;
    color: #858585;
    padding: 2px 5px;
    font-size: 0.8rem;
}

.file-action-btn:hover {
    color: #d4d4d4;
}

.wasm-section {
    margin-top: 20px;
    border-top: 1px solid #3e3e42;
    padding-top: 20px;
}

.wasm-info {
    background-color: #2a2d2e;
    padding: 10px;
    border-radius: 4px;
    margin-bottom: 15px;
    font-size: 0.85rem;
}

.wasm-info.warning {
    background-color: #5c2d0c;
    border-left: 3px solid #f5d76e;
}

.wasm-info.success {
    background-color: #2d5c0c;
    border-left: 3px solid #3fa33a;
}

.upload-area {
    border: 2px dashed #3e3e42;
    border-radius: 8px;
    padding: 20px;
    text-align: center;
```

```
    cursor: pointer;
    transition: border-color 0.2s;
    margin-bottom: 15px;
}

.upload-area:hover {
    border-color: #0e639c;
}

.upload-area.dragover {
    border-color: #569cd6;
    background-color: rgba(86, 156, 214, 0.1);
}

.upload-icon {
    font-size: 2rem;
    color: #569cd6;
    margin-bottom: 10px;
}

.upload-area p {
    margin-bottom: 5px;
}

.upload-hint {
    font-size: 0.8rem;
    color: #858585;
}

.wasm-list {
    list-style-type: none;
    margin-top: 10px;
    max-height: 150px;
    overflow-y: auto;
}

.wasm-list li {
    padding: 8px 12px;
    background-color: #2a2d2e;
    border-radius: 4px;
    margin-bottom: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
```

```
    font-size: 0.9rem;
}

.wasm-size {
    color: #858585;
    font-size: 0.8rem;
}

.editor-container {
    flex: 1;
    display: flex;
    flex-direction: column;
    overflow: hidden;
}

.editor-tabs {
    display: flex;
    background-color: #2d2d30;
    border-bottom: 1px solid #3e3e42;
    overflow-x: auto;
}

.tab {
    padding: 10px 20px;
    border-right: 1px solid #3e3e42;
    cursor: pointer;
    white-space: nowrap;
    display: flex;
    align-items: center;
    gap: 8px;
}

.tab:hover {
    background-color: #2a2d2e;
}

.tab.active {
    background-color: #1e1e1e;
}

.close-tab {
    margin-left: 5px;
    opacity: 0.7;
}
```

```
.close-tab:hover {
  opacity: 1;
}

#editor {
  flex: 1;
  overflow: hidden;
}

.config-panel {
  margin-top: 20px;
}

.config-item {
  margin-bottom: 15px;
}

label {
  display: block;
  margin-bottom: 5px;
  font-size: 0.9rem;
  color: #cccccc;
}

input[type="text"], select {
  width: 100%;
  padding: 8px 12px;
  background-color: #3e3e42;
  border: 1px solid #565656;
  border-radius: 4px;
  color: #d4d4d4;
}

input[type="text"]:focus, select:focus {
  outline: none;
  border-color: #007acc;
}

.checkbox-group {
  display: flex;
  align-items: center;
  gap: 8px;
  margin-top: 5px;
```

```
}

input[type="checkbox"] {
  accent-color: #0e639c;
}

.publish-modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.8);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1000;
  display: none;
}

.modal-content {
  background-color: #252526;
  border-radius: 8px;
  padding: 25px;
  width: 90%;
  max-width: 600px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.5);
  max-height: 80vh;
  overflow-y: auto;
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.modal-header h2 {
  color: #569cd6;
  font-size: 1.5rem;
}

.close-modal {
```

```
    font-size: 1.8rem;
    cursor: pointer;
    color: #cccccc;
}

.close-modal:hover {
    color: white;
}

.publish-result {
    background-color: #1e1e1e;
    padding: 15px;
    border-radius: 4px;
    margin-top: 20px;
    font-family: monospace;
    font-size: 0.9rem;
    white-space: pre-wrap;
    word-break: break-all;
    display: none;
}

.result-item {
    margin-bottom: 10px;
}

.result-label {
    color: #9cdcfe;
    font-weight: bold;
}

.result-value {
    color: #ce9178;
}

.result-value a {
    color: #569cd6;
    text-decoration: none;
}

.result-value a:hover {
    text-decoration: underline;
}

.loading {
```

```
display: none;
text-align: center;
margin: 20px 0;
color: #569cd6;
}

.spinner {
border: 3px solid rgba(86, 156, 214, 0.3);
border-radius: 50%;
border-top: 3px solid #569cd6;
width: 30px;
height: 30px;
animation: spin 1s linear infinite;
margin: 0 auto 10px;
}

@keyframes spin {
0% { transform: rotate(0deg); }
100% { transform: rotate(360deg); }
}

.notification {
position: fixed;
bottom: 20px;
right: 20px;
padding: 12px 20px;
background-color: #0c7b93;
color: white;
border-radius: 4px;
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
z-index: 1001;
transform: translateY(100px);
opacity: 0;
transition: transform 0.3s, opacity 0.3s;
}

.notification.show {
transform: translateY(0);
opacity: 1;
}

.notification.error {
background-color: #a12c2c;
}
```

```
.notification.success {
  background-color: #388a34;
}

footer {
  padding: 10px 20px;
  background-color: #252526;
  border-top: 1px solid #3e3e42;
  font-size: 0.8rem;
  color: #858585;
  text-align: center;
}

.field-row {
  display: flex;
  gap: 10px;
  margin-bottom: 15px;
}

.field-row .config-item {
  flex: 1;
  margin-bottom: 0;
}

.advanced-options {
  background-color: #2a2d2e;
  padding: 15px;
  border-radius: 4px;
  margin-top: 15px;
  border-left: 3px solid #569cd6;
}

.advanced-options h4 {
  margin-bottom: 10px;
  color: #569cd6;
}

.option-description {
  font-size: 0.85rem;
  color: #858585;
  margin-top: 5px;
}
```

```

.type-info {
    font-size: 0.8rem;
    padding: 8px;
    border-radius: 4px;
    margin-top: 5px;
    display: none;
}

.basic-info {
    background-color: #2d5c0c;
    border-left: 3px solid #3fa33a;
}

.standard-info {
    background-color: #5c5c0c;
    border-left: 3px solid #f5d76e;
}

.advanced-info {
    background-color: #5c2d0c;
    border-left: 3px solid #d16969;
}
</style>
</head>
<body>
    <header>
        <div class="logo">Pack Editor</div>
        <div class="controls">
            <button id="newFileBtn" class="secondary">New File</button>
            <button id="saveBtn">Save</button>
            <button id="publishBtn" class="success">Publish Pack</button>
        </div>
    </header>

    <div class="container">
        <div class="sidebar">
            <h3>Files</h3>
            <ul class="file-list" id="fileList">
                <!-- Files will be added here dynamically -->
            </ul>

            <div class="wasm-section">
                <h3>WebAssembly Files</h3>

```

```
<div class="wasm-info warning" id="wasmInfo">
  <strong>Important:</strong> WASM files cannot be edited directly. You can only
  upload valid WebAssembly binary files (.wasm). These will be served from the CDN.
</div>
```

```
<div class="upload-area" id="uploadArea">
  <div class="upload-icon">⬆</div>
  <p>Drag & drop WASM files here</p>
  <p class="upload-hint">or click to browse</p>
  <p class="upload-hint">Only .wasm files accepted</p>
</div>
```

```
<ul class="wasm-list" id="wasmList">
  <!-- WASM files will be listed here -->
</ul>
</div>
```

```
<div class="config-panel">
  <h3>Pack Configuration</h3>
```

```
  <div class="config-item">
    <label for="packName">Pack Name</label>
    <input type="text" id="packName" placeholder="my-awesome-pack"
value="my-awesome-pack">
  </div>
```

```
<div class="field-row">
  <div class="config-item">
    <label for="packType">Package Type</label>
    <select id="packType">
      <option value="basic">Basic</option>
      <option value="standard">Standard</option>
      <option value="advanced">Advanced</option>
    </select>
    <div id="typeInfo" class="type-info"></div>
  </div>
```

```
  <div class="config-item">
    <label for="version">Version</label>
    <input type="text" id="version" placeholder="1.0.0" value="1.0.0">
  </div>
</div>
```

```
<div class="config-item">
```

```

        <label>Visibility</label>
        <div class="checkbox-group">
            <input type="checkbox" id="isPublic" checked>
            <label for="isPublic">Public Package</label>
        </div>
    </div>

    <div class="advanced-options">
        <h4>Advanced Options</h4>

        <div class="config-item">
            <div class="checkbox-group">
                <input type="checkbox" id="isNewVersion">
                <label for="isNewVersion">Create as new version</label>
            </div>
            <div class="option-description">Check this if you're creating a new version of an
existing package</div>
        </div>

        <div id="newVersionFields" style="display: none;">
            <div class="config-item">
                <label for="basePackId">Base Pack ID</label>
                <input type="text" id="basePackId" placeholder="Existing pack ID">
                <div class="option-description">ID of the package you're creating a new
version for</div>
            </div>

            <div class="config-item">
                <label for="editToken">Edit Token</label>
                <input type="text" id="editToken" placeholder="Edit token (if required)">
                <div class="option-description">Required if you're not the original
publisher</div>
            </div>
        </div>

        <div class="config-item">
            <label for="userId">User ID (Optional)</label>
            <input type="text" id="userId" placeholder="Your user ID">
            <div class="option-description">For collaboration and version tracking</div>
        </div>
    </div>
</div>
</div>

```

```

<div class="editor-container">
  <div class="editor-tabs" id="editorTabs">
    <!-- Editor tabs will be added here dynamically -->
  </div>
  <div id="editor"></div>
</div>
</div>

<div class="publish-modal" id="publishModal">
  <div class="modal-content">
    <div class="modal-header">
      <h2>Publish Pack</h2>
      <div class="close-modal" id="closeModal">&times;</div>
    </div>

    <p>Ready to publish your pack? This will upload all files and generate a unique URL for
your package.</p>

    <div id="validationResults">
      <!-- Validation results will appear here -->
    </div>

    <div class="loading" id="publishLoading">
      <div class="spinner"></div>
      <p>Publishing your pack...</p>
    </div>

    <div class="publish-result" id="publishResult">
      <!-- Publish results will be displayed here -->
    </div>

    <div style="display: flex; gap: 10px; margin-top: 20px;">
      <button id="confirmPublishBtn" class="success" style="flex: 1;">Publish
Now</button>
      <button id="cancelPublishBtn" class="secondary">Cancel</button>
    </div>
  </div>
</div>

<div class="notification" id="notification">Message</div>

<footer>
  Pack Editor - Edit and publish packages with WebAssembly support
</footer>

```

```

<!-- Monaco Editor loader -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/loader.min.js"></script>
<script>
    // Monaco editor configuration
    require.config({ paths: { vs:
'https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs' } });

    // Initialize app state with new fields
    const appState = {
        files: {
            'package.json': {
                name: 'package.json',
                content: JSON.stringify({
                    name: 'my-awesome-pack',
                    version: '1.0.0',
                    type: 'module',
                    description: 'My awesome package with WASM support',
                    main: 'index.js',
                    author: "",
                    license: 'MIT'
                }, null, 2),
                language: 'json',
                editable: true
            },
            'index.js': {
                name: 'index.js',
                content: `// Main entry point for your package

export function helloWorld() {
    console.log('Hello from my awesome pack!');
    return 'Hello World!';
}

// Add your code here
export const version = '1.0.0';

// Example utility function
export function sum(a, b) {
    return a + b;
}`,
                language: 'javascript',
                editable: true
            }
        }
    };

```

```
    },  
    'README.md': {  
      name: 'README.md',  
      content: `# My Awesome Pack`  
    }  
  },  
  wasmFiles: {},  
  activeFile: 'package.json',  
  editor: null,  
  packName: 'my-awesome-pack',  
  packType: 'basic',  
  version: '1.0.0',  
  isPublic: true,  
}
```

This is a sample package created with Pack Editor.

Installation

```
```bash  
pack install my-awesome-pack
```
```

Usage

```
```javascript  
import { helloWorld } from 'my-awesome-pack';

helloWorld(); // Outputs: Hello from my awesome pack!
```
```

API

helloWorld()
Prints a greeting message and returns "Hello World!"

sum(a, b)
Returns the sum of two numbers

License

MIT

```
,  
  {  
    language: 'markdown',  
    editable: true  
  }  
},  
wasmFiles: {},  
activeFile: 'package.json',  
editor: null,  
packName: 'my-awesome-pack',  
packType: 'basic',  
version: '1.0.0',  
isPublic: true,  
}
```

```
    isNewVersion: false,
    basePackId: "",
    userId: "",
    editToken: ""
  };

// Package type limits from API
const PACKAGE_TYPE_LIMITS = {
  basic: {
    maxFiles: 20,
    maxSize: 5 * 1024 * 1024, // 5MB
    description: 'Simple packages with basic JS, no Node.js modules'
  },
  standard: {
    maxFiles: 50,
    maxSize: 10 * 1024 * 1024, // 10MB
    description: 'Standard packages with limited Node.js modules'
  },
  advanced: {
    maxFiles: 100,
    maxSize: 25 * 1024 * 1024, // 25MB
    description: 'Advanced packages with full Node.js module support'
  }
};
```

```
// DOM Elements
const fileList = document.getElementById('fileList');
const editorTabs = document.getElementById('editorTabs');
const editorContainer = document.getElementById('editor');
const packNameInput = document.getElementById('packName');
const packTypeSelect = document.getElementById('packType');
const versionInput = document.getElementById('version');
const isPublicCheckbox = document.getElementById('isPublic');
const isNewVersionCheckbox = document.getElementById('isNewVersion');
const newVersionFields = document.getElementById('newVersionFields');
const basePackIdInput = document.getElementById('basePackId');
const editTokenInput = document.getElementById('editToken');
const userIdInput = document.getElementById('userId');
const typeInfo = document.getElementById('typeInfo');
const publishModal = document.getElementById('publishModal');
const closeModal = document.getElementById('closeModal');
const cancelPublishBtn = document.getElementById('cancelPublishBtn');
const confirmPublishBtn = document.getElementById('confirmPublishBtn');
const publishResult = document.getElementById('publishResult');
```

```

const publishLoading = document.getElementById('publishLoading');
const notification = document.getElementById('notification');
const uploadArea = document.getElementById('uploadArea');
const wasmList = document.getElementById('wasmList');
const wasmInfo = document.getElementById('wasmInfo');
const validationResults = document.getElementById('validationResults');

// File input for WASM uploads (hidden)
const fileInput = document.createElement('input');
fileInput.type = 'file';
fileInput.accept = '.wasm';
fileInput.multiple = true;
fileInput.style.display = 'none';
document.body.appendChild(fileInput);

// Initialize Monaco Editor
require(['vs/editor/editor.main'], function() {
  // Create editor instance
  appState.editor = monaco.editor.create(editorContainer, {
    value: appState.files['package.json'].content,
    language: 'json',
    theme: 'vs-dark',
    minimap: { enabled: true },
    scrollBeyondLastLine: false,
    automaticLayout: true,
    fontSize: 14,
    lineNumbers: 'on',
    roundedSelection: false,
    scrollbar: {
      vertical: 'auto',
      horizontal: 'auto'
    },
    readOnly: false
  });

  // Update app state when editor content changes
  appState.editor.onDidChangeModelContent(() => {
    const activeFile = appState.activeFile;
    if (activeFile && appState.files[activeFile]) {
      appState.files[activeFile].content = appState.editor.getValue();

      // If this is package.json, update the pack name if changed
      if (activeFile === 'package.json') {
        try {

```

```

const packageJson = JSON.parse(appState.editor.getValue());
if (packageJson.name && packageJson.name !== appState.packName) {
  appState.packName = packageJson.name;
  packNameInput.value = packageJson.name;
}
if (packageJson.version && packageJson.version !== appState.version) {
  appState.version = packageJson.version;
  versionInput.value = packageJson.version;
}
if (packageJson.type && packageJson.type !== appState.packType) {
  // Map module/library/template/plugin to basic/standard/advanced
  if (packageJson.type === 'module' || packageJson.type === 'library') {
    appState.packType = 'standard';
  } else if (packageJson.type === 'template' || packageJson.type === 'plugin')
{
    appState.packType = 'advanced';
  } else {
    appState.packType = packageJson.type;
  }
  packTypeSelect.value = appState.packType;
  updateTypeInfo();
}
} catch (e) {
  // Invalid JSON, ignore
}
}
}
});

// Initialize UI
renderFileList();
renderEditorTabs();
setupEventListeners();
updateTypeInfo();
});

// Render the file list in the sidebar
function renderFileList() {
  fileList.innerHTML = "";

  Object.keys(appState.files).forEach(fileName => {
    const file = appState.files[fileName];
    const li = document.createElement('li');

```

```

    li.className = `file-item ${fileName === appState.activeFile ? 'active' : ''}
    ${file.language}-file`;

    const fileExt = fileName.split('.').pop();
    const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

    li.innerHTML = `
      <div>
        <span class="file-icon">${icon}</span>
        <span>${fileName}</span>
      </div>
      <div class="file-actions">
        <button class="file-action-btn" data-file="${fileName}"
title="Rename">✏️</button>
        ${fileName !== 'package.json' && fileName !== 'index.js' ?
        `<button class="file-action-btn" data-file="${fileName}"
title="Delete">🗑️</button>` : ''}
      </div>
    `;

    li.addEventListener('click', (e) => {
      if (!e.target.closest('.file-action-btn')) {
        switchFile(fileName);
      }
    });

    fileList.appendChild(li);
  });

  // Add event listeners for file actions
  document.querySelectorAll('.file-action-btn').forEach(btn => {
    btn.addEventListener('click', (e) => {
      e.stopPropagation();
      const fileName = btn.dataset.file;

      if (btn.textContent.includes('✏️')) {
        renameFile(fileName);
      } else if (btn.textContent.includes('🗑️')) {
        deleteFile(fileName);
      }
    });
  });
}

```

```

// Render WASM file list
function renderWasmList() {
  wasmList.innerHTML = "";

  if (Object.keys(appState.wasmFiles).length === 0) {
    const li = document.createElement('li');
    li.textContent = 'No WASM files uploaded';
    li.style.color = '#858585';
    li.style.fontStyle = 'italic';
    wasmList.appendChild(li);
    wasmInfo.className = 'wasm-info warning';
    wasmInfo.innerHTML = '<strong>Important:</strong> WASM files cannot be edited
directly. You can only upload valid WebAssembly binary files (.wasm). These will be served from
the CDN.';
  } else {
    Object.keys(appState.wasmFiles).forEach(fileName => {
      const file = appState.wasmFiles[fileName];
      const li = document.createElement('li');

      li.innerHTML = `
        <div>
          <span class="file-icon">⚡ </span>
          <span>${fileName}</span>
        </div>
        <div>
          <span class="wasm-size">${formatFileSize(file.size)}</span>
          <button class="file-action-btn" data-wasm="${fileName}"
title="Remove">🗑️ </button>
        </div>
      `;

      wasmList.appendChild(li);
    });

    wasmInfo.className = 'wasm-info success';
    wasmInfo.innerHTML = `<strong>✓ Ready:</strong>
${Object.keys(appState.wasmFiles).length} WASM file(s) uploaded and validated.`;

    // Add event listeners for remove buttons
    document.querySelectorAll('[data-wasm]').forEach(btn => {
      btn.addEventListener('click', (e) => {
        const fileName = btn.dataset.wasm;
        deleteWasmFile(fileName);
      });
    });
  }
}

```

```

    });
  }
}

// Render editor tabs
function renderEditorTabs() {
  editorTabs.innerHTML = "";

  Object.keys(appState.files).forEach(fileName => {
    const tab = document.createElement('div');
    tab.className = `tab ${fileName === appState.activeFile ? 'active' : ''}`;
    tab.dataset.file = fileName;

    const fileExt = fileName.split('.').pop();
    const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

    tab.innerHTML = `
      <span class="file-icon">${icon}</span>
      <span>${fileName}</span>
      ${fileName !== 'package.json' && fileName !== 'index.js' ?
        '<span class="close-tab" data-file="' + fileName + '">×</span>' : ''}
    `;

    tab.addEventListener('click', (e) => {
      if (!e.target.classList.contains('close-tab')) {
        switchFile(fileName);
      }
    });

    editorTabs.appendChild(tab);
  });

  // Add event listeners for close tab buttons
  document.querySelectorAll('.close-tab').forEach(btn => {
    btn.addEventListener('click', (e) => {
      e.stopPropagation();
      const fileName = btn.dataset.file;
      deleteFile(fileName);
    });
  });
}

// Switch to a different file
function switchFile(fileName) {

```

```
if (!appState.files[fileName]) return;

appState.activeFile = fileName;
const file = appState.files[fileName];

// Update editor content and language
appState.editor.setValue(file.content);
monaco.editor.setModelLanguage(appState.editor.getModel(), file.language);

// Update UI
renderFileList();
renderEditorTabs();
}

// Create a new file
function createNewFile() {
  const fileName = prompt('Enter file name (with extension, e.g., utils.js):', 'newfile.js');
  if (!fileName) return;

  if (appState.files[fileName] || appState.wasmFiles[fileName]) {
    showNotification('File already exists!', 'error');
    return;
  }

  const fileExt = fileName.split('.').pop();
  let language = 'plaintext';
  let content = '';

  switch (fileExt) {
    case 'js':
      language = 'javascript';
      content = `// ${fileName}\n\n`;
      break;
    case 'json':
      language = 'json';
      content = '{\n  \n}\n';
      break;
    case 'md':
      language = 'markdown';
      content = `# ${fileName}\n\n`;
      break;
    case 'html':
      language = 'html';
```

```
        content = `<!DOCTYPE html>\n<html>\n<head>\n<title></title>\n</head>\n<body>\n  \n</body>\n</html>\n`;
```

```
        break;
```

```
    case 'css':
```

```
        language = 'css';
```

```
        content = `/* ${fileName} */\n\n`;
```

```
        break;
```

```
    case 'py':
```

```
        language = 'python';
```

```
        content = `# ${fileName}\n\n`;
```

```
        break;
```

```
    case 'ts':
```

```
    case 'tsx':
```

```
        language = 'typescript';
```

```
        content = `// ${fileName}\n\n`;
```

```
        break;
```

```
    }
```

```
    appState.files[fileName] = {
```

```
        name: fileName,
```

```
        content,
```

```
        language,
```

```
        editable: true
```

```
    };
```

```
    switchFile(fileName);
```

```
    showNotification(`Created ${fileName}`, 'success');
```

```
  }
```

```
// Rename a file
```

```
function renameFile(oldName) {
```

```
    const newName = prompt('Enter new file name:', oldName);
```

```
    if (!newName || newName === oldName) return;
```

```
    if (appState.files[newName] || appState.wasmFiles[newName]) {
```

```
        showNotification('File already exists!', 'error');
```

```
        return;
```

```
    }
```

```
    appState.files[newName] = {
```

```
        ...appState.files[oldName],
```

```
        name: newName
```

```
    };
```

```

    delete appState.files[oldName];

    if (appState.activeFile === oldName) {
      appState.activeFile = newName;
    }

    renderFileList();
    renderEditorTabs();
    showNotification(`Renamed ${oldName} to ${newName}`, 'success');
  }

  // Delete a file
  function deleteFile(fileName) {
    if (fileName === 'package.json' || fileName === 'index.js') {
      showNotification('Cannot delete required files!', 'error');
      return;
    }

    if (confirm(`Are you sure you want to delete ${fileName}?`)) {
      delete appState.files[fileName];

      if (appState.activeFile === fileName) {
        const remainingFiles = Object.keys(appState.files);
        if (remainingFiles.length > 0) {
          switchFile(remainingFiles[0]);
        }
      }

      renderFileList();
      renderEditorTabs();
      showNotification(`Deleted ${fileName}`, 'success');
    }
  }

  // Handle WASM file upload
  async function handleWasmUpload(files) {
    for (const file of files) {
      if (!file.name.endsWith('.wasm')) {
        showNotification(`Skipped ${file.name}: Not a .wasm file`, 'error');
        continue;
      }

      // Validate WASM file
      const isValid = await validateWasmFile(file);
    }
  }

```

```

    if (!isValid) {
      showNotification(`Skipped ${file.name}: Invalid WebAssembly file`, 'error');
      continue;
    }

    // Read file as ArrayBuffer
    const reader = new FileReader();
    reader.onload = async (e) => {
      const arrayBuffer = e.target.result;

      // Store WASM file
      appState.wasmFiles[file.name] = {
        name: file.name,
        size: file.size,
        data: arrayBuffer
      };

      renderWasmList();
      showNotification(`Uploaded ${file.name} (${formatFileSize(file.size)})`, 'success');
    };

    reader.readAsArrayBuffer(file);
  }
}

// Validate a WASM file
async function validateWasmFile(file) {
  try {
    // Basic WASM validation - check magic number
    const arrayBuffer = await file.slice(0, 8).arrayBuffer();
    const header = new Uint8Array(arrayBuffer);

    // WASM magic number: \0asm
    const wasmMagic = [0x00, 0x61, 0x73, 0x6D];

    for (let i = 0; i < 4; i++) {
      if (header[i] !== wasmMagic[i]) {
        return false;
      }
    }

    return true;
  } catch (error) {
    console.error('WASM validation error:', error);
  }
}

```

```

        return false;
    }
}

// Delete a WASM file
function deleteWasmFile(fileName) {
    if (confirm(`Are you sure you want to remove ${fileName}?`)) {
        delete appState.wasmFiles[fileName];
        renderWasmList();
        showNotification(`Removed ${fileName}`, 'success');
    }
}

// Format file size
function formatFileSize(bytes) {
    if (bytes === 0) return '0 Bytes';
    const k = 1024;
    const sizes = ['Bytes', 'KB', 'MB', 'GB'];
    const i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i];
}

// Update package type info display
function updateTypeInfo() {
    const limits = PACKAGE_TYPE_LIMITS[appState.packType];
    if (limits) {
        typeInfo.textContent = `${limits.description}. Max ${limits.maxFiles} files,
${limits.maxSize / 1024 / 1024}MB total.`;
        typeInfo.className = `type-info ${appState.packType}-info`;
        typeInfo.style.display = 'block';
    }
}

// Show notification
function showNotification(message, type = 'info') {
    notification.textContent = message;
    notification.className = `notification ${type}`;
    notification.classList.add('show');

    setTimeout(() => {
        notification.classList.remove('show');
    }, 3000);
}

```

```

// Validate package before publishing
function validatePackage() {
  const errors = [];
  const warnings = [];

  // Check package.json
  try {
    const packageJson = JSON.parse(appState.files['package.json'].content);

    if (!packageJson.name || packageJson.name.trim() === "") {
      errors.push('Package name is required in package.json');
    }

    if (!packageJson.version) {
      warnings.push('Consider adding a version field to package.json');
    }

    if (!packageJson.main) {
      warnings.push('Consider specifying a main entry point in package.json');
    }
  } catch (e) {
    errors.push('Invalid JSON in package.json');
  }

  // Check if index.js exists
  if (!appState.files['index.js']) {
    warnings.push('index.js is recommended as the main entry point');
  }

  // Check for invalid file names
  Object.keys(appState.files).forEach(fileName => {
    if (fileName.includes(' ')) {
      warnings.push(`File "${fileName}" contains spaces, consider using hyphens or
underscores`);
    }
  });

  // Check package type limits
  const limits = PACKAGE_TYPE_LIMITS[appState.packType];
  const totalFiles = Object.keys(appState.files).length +
Object.keys(appState.wasmFiles).length;

  if (totalFiles > limits.maxFiles) {

```

```

        errors.push(`Package type "${appState.packType}" allows maximum
        ${limits.maxFiles} files, but you have ${totalFiles}`);
    }

    // Check if creating new version without base pack ID
    if (appState.isNewVersion && !appState.basePackId) {
        errors.push('Base Pack ID is required when creating a new version');
    }

    return { errors, warnings };
}

// Publish package to API - UPDATED FOR NEW API
async function publishPackage() {
    // Validate package first
    const validation = validatePackage();

    if (validation.errors.length > 0) {
        showNotification('Cannot publish: ' + validation.errors[0], 'error');
        return;
    }

    // Show loading state
    publishLoading.style.display = 'block';
    validationResults.innerHTML = "";

    try {
        // Prepare package.json
        const packageJson = JSON.parse(appState.files['package.json'].content);
        packageJson.name = appState.packName;

        // Prepare all files for upload
        const files = {};

        // Add regular files
        Object.keys(appState.files).forEach(fileName => {
            files[fileName] = appState.files[fileName].content;
        });

        // Add WASM files - convert ArrayBuffer to base64 string
        Object.keys(appState.wasmFiles).forEach(fileName => {
            const wasmFile = appState.wasmFiles[fileName];
            // Convert ArrayBuffer to base64 string
            const bytes = new Uint8Array(wasmFile.data);

```

```
    let binary = "";
    for (let i = 0; i < bytes.byteLength; i++) {
        binary += String.fromCharCode(bytes[i]);
    }
    files[fileName] = btoa(binary);
});
```

```
// Prepare request body for new API
const requestBody = {
    name: appState.packName,
    packJson: JSON.stringify(packageJson),
    files: files,
    isPublic: appState.isPublic,
    packageType: appState.packageType,
    version: appState.version,
    isNewVersion: appState.isNewVersion,
    basePackId: appState.basePackId || null,
    userId: appState.userId || null,
    editToken: appState.editToken || null,
    collaborators: [] // Can be extended later
};
```

```
console.log('Sending to API:', {
    name: requestBody.name,
    packageType: requestBody.packageType,
    version: requestBody.version,
    isNewVersion: requestBody.isNewVersion,
    fileCount: Object.keys(requestBody.files).length,
    isPublic: requestBody.isPublic
});
```

```
// Call publish API
const response = await fetch('/api/publish.js', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestBody)
});
```

```
const responseText = await response.text();
console.log('API Response:', responseText);
```

```
if (!response.ok) {
```

```
        throw new Error(`HTTP ${response.status}: ${response.statusText}. Response:
        ${responseText}`);
    }
}
```

```
const result = JSON.parse(responseText);
```

```
// Hide loading, show results
publishLoading.style.display = 'none';
publishResult.style.display = 'block';
```

```
// Display results
publishResult.innerHTML = `
    <div class="result-item">
        <span class="result-label">✓ Published successfully!</span>
    </div>
    <div class="result-item">
        <span class="result-label">Package Type:</span>
        <span class="result-value">${appState.packType}</span>
    </div>
    <div class="result-item">
        <span class="result-label">Version:</span>
        <span class="result-value">${result.version}</span>
    </div>
    <div class="result-item">
        <span class="result-label">Pack ID:</span>
        <span class="result-value">${result.packId}</span>
    </div>
    <div class="result-item">
        <span class="result-label">URL ID:</span>
        <span class="result-value">${result.urlId}</span>
    </div>
    <div class="result-item">
        <span class="result-label">CDN URL:</span>
        <span class="result-value"><a href="${result.cdnUrl}"
target="_blank">${result.cdnUrl}</a></span>
    </div>
    <div class="result-item">
        <span class="result-label">Worker URL:</span>
        <span class="result-value"><a href="${result.workerUrl}"
target="_blank">${result.workerUrl}</a></span>
    </div>
    <div class="result-item">
        <span class="result-label">Install Command:</span>
        <span class="result-value">${result.installCommand}</span>
    </div>
`
```

```

    </div>
    ${result.encryptedKey ? `
    <div class="result-item">
      <span class="result-label">Encryption Key:</span>
      <span class="result-value">${result.encryptedKey}</span>
    </div>
    <div class="result-item">
      <span class="result-label">⚠ Save this key! It won't be shown again.</span>
    </div>
    ` : ""}
    ${result.isNewVersion ? `
    <div class="result-item">
      <span class="result-label">New Version:</span>
      <span class="result-value">✔ Created new version from base pack
    </div>
    ` : ""}
    <div class="result-item">
      <span class="result-label">Advanced Features:</span>
      <span class="result-value">Versioning: ${result.advancedFeatures?.versioning ?
    '✔' : '✗'},
      Collaboration: ${result.advancedFeatures?.collaboration ? '✔' : '✗'}</span>
    </div>
    <div class="result-item">
      <button id="exploreBtn" class="success" style="width: 100%; margin-top:
    10px;">Explore Package</button>
    </div>
  `;

  // Add event listener for explore button
  document.getElementById('exploreBtn').addEventListener('click', () => {
    window.location.href = 'explore.html';
  });

  } catch (error) {
    publishLoading.style.display = 'none';
    showNotification(`Publish failed: ${error.message}`, 'error');
    console.error('Publish error:', error);
  }
}

// Setup event listeners
function setupEventListeners() {
  // Pack configuration

```

```
packNameInput.addEventListener('input', (e) => {
  appState.packName = e.target.value;
});

packTypeSelect.addEventListener('change', (e) => {
  appState.packType = e.target.value;
  updateTypeInfo();
});

versionInput.addEventListener('input', (e) => {
  appState.version = e.target.value;
});

isPublicCheckbox.addEventListener('change', (e) => {
  appState.isPublic = e.target.checked;
});

isNewVersionCheckbox.addEventListener('change', (e) => {
  appState.isNewVersion = e.target.checked;
  newVersionFields.style.display = e.target.checked ? 'block' : 'none';
});

basePackIdInput.addEventListener('input', (e) => {
  appState.basePackId = e.target.value;
});

editTokenInput.addEventListener('input', (e) => {
  appState.editToken = e.target.value;
});

userIdInput.addEventListener('input', (e) => {
  appState.userId = e.target.value;
});

// New file button
document.getElementById('newFileBtn').addEventListener('click', createNewFile);

// Save button
document.getElementById('saveBtn').addEventListener('click', () => {
  showNotification('All changes saved locally', 'success');
});

// Publish button
document.getElementById('publishBtn').addEventListener('click', () => {
```

```

// Validate before showing modal
const validation = validatePackage();

validationResults.innerHTML = "";

if (validation.errors.length > 0) {
  validationResults.innerHTML = `
    <div style="background-color: #5c2d0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
      <strong>Errors:</strong>
      <ul style="margin: 5px 0 0 20px;">
        ${validation.errors.map(err => `<li>${err}</li>`).join("")}
      </ul>
    </div>
  `;
} else if (validation.warnings.length > 0) {
  validationResults.innerHTML = `
    <div style="background-color: #5c5c0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
      <strong>Warnings:</strong>
      <ul style="margin: 5px 0 0 20px;">
        ${validation.warnings.map(warn => `<li>${warn}</li>`).join("")}
      </ul>
    </div>
  `;
}

// Add package type info to modal
const limits = PACKAGE_TYPE_LIMITS[appState.packType];
const totalFiles = Object.keys(appState.files).length +
Object.keys(appState.wasmFiles).length;

validationResults.innerHTML += `
  <div style="background-color: #2a2d2e; padding: 10px; border-radius: 4px; margin:
10px 0;">
    <strong>Package Type:</strong> ${appState.packType}
    <div style="font-size: 0.9rem; margin-top: 5px;">
      ${limits.description}<br>
      Files: ${totalFiles} / ${limits.maxFiles}<br>
      Max size: ${limits.maxSize / 1024 / 1024}MB
    </div>
  </div>
  `;

```

```

        publishModal.style.display = 'flex';
        publishResult.style.display = 'none';
        publishLoading.style.display = 'none';
    });

    // Modal controls
    closeModal.addEventListener('click', () => {
        publishModal.style.display = 'none';
    });

    cancelPublishBtn.addEventListener('click', () => {
        publishModal.style.display = 'none';
    });

    confirmPublishBtn.addEventListener('click', publishPackage);

    // WASM file upload
    uploadArea.addEventListener('click', () => {
        fileInput.click();
    });

    fileInput.addEventListener('change', (e) => {
        handleWasmUpload(Array.from(e.target.files));
        fileInput.value = "";
    });

    // Drag and drop for WASM files
    uploadArea.addEventListener('dragover', (e) => {
        e.preventDefault();
        uploadArea.classList.add('dragover');
    });

    uploadArea.addEventListener('dragleave', () => {
        uploadArea.classList.remove('dragover');
    });

    uploadArea.addEventListener('drop', (e) => {
        e.preventDefault();
        uploadArea.classList.remove('dragover');

        const files = Array.from(e.dataTransfer.files);
        handleWasmUpload(files);
    });
}

```

```

// Initialize WASM list
renderWasmList();

// Close modal when clicking outside
window.addEventListener('click', (e) => {
  if (e.target === publishModal) {
    publishModal.style.display = 'none';
  }
});
</script>
</body>
</html>

```

```

public/Explore/explore.html
<!DOCTYPE html>
<html>
<head>
  <title>Explore Packs - PackCDN</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body { margin: 0; font-family: -apple-system, BlinkMacSystemFont, sans-serif; background:
    #f8f9fa; }
    .container { max-width: 1400px; margin: 0 auto; padding: 20px; }
    header { background: white; padding: 30px; border-radius: 12px; margin-bottom: 30px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.08); }
    .search-box { display: flex; gap: 10px; margin-top: 20px; }
    .search-box input { flex: 1; padding: 14px; border: 2px solid #e2e8f0; border-radius: 8px;
    font-size: 16px; transition: border-color 0.2s; }
    .search-box input:focus { outline: none; border-color: #667eea; }
    .search-box button { padding: 14px 36px; background: linear-gradient(135deg, #667eea 0%,
    #764ba2 100%); color: white; border: none; border-radius: 8px; cursor: pointer; font-size: 16px;
    font-weight: 600; }
    .advanced-search { margin-top: 20px; padding: 20px; background: #f8f9ff; border-radius: 8px;
    display: none; }
    .advanced-search.active { display: block; }
    .filter-row { display: flex; flex-wrap: wrap; gap: 15px; margin-bottom: 15px; }
    .filter-group { flex: 1; min-width: 200px; }
    .filter-group label { display: block; margin-bottom: 8px; color: #4a5568; font-weight: 500;
    font-size: 14px; }
    .filter-select, .filter-input { width: 100%; padding: 10px; border: 1px solid #e2e8f0;
    border-radius: 6px; background: white; }
    .toggle-advanced { background: none; border: none; color: #667eea; cursor: pointer;
    font-size: 14px; margin-top: 10px; }

```

```
.filters { display: flex; gap: 10px; margin-top: 20px; flex-wrap: wrap; }
.filter-btn { padding: 8px 20px; background: white; border: 2px solid #e2e8f0; border-radius: 20px; cursor: pointer; transition: all 0.2s; font-weight: 500; }
.filter-btn.active { background: #667eea; color: white; border-color: #667eea; }
.stats-banner { display: flex; gap: 30px; background: linear-gradient(135deg, #f6f8ff 0%, #f0f4ff 100%); padding: 20px; border-radius: 8px; margin: 20px 0; }
.stat-item { text-align: center; flex: 1; }
.stat-value { font-size: 24px; font-weight: bold; color: #667eea; }
.stat-label { color: #718096; font-size: 14px; margin-top: 5px; }
.packs-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(340px, 1fr)); gap: 25px; margin-top: 30px; }
.pack-card { background: white; border-radius: 12px; padding: 24px; box-shadow: 0 4px 12px rgba(0,0,0,0.08); cursor: pointer; transition: all 0.3s ease; border: 2px solid transparent; position: relative; }
.pack-card:hover { transform: translateY(-5px); box-shadow: 0 8px 24px rgba(0,0,0,0.12); border-color: #667eea; }
.pack-card.verified::before { content: "✓ Verified"; position: absolute; top: -10px; right: 15px; background: #10b981; color: white; padding: 4px 12px; border-radius: 20px; font-size: 11px; font-weight: 600; }
.pack-header { display: flex; justify-content: space-between; align-items: start; margin-bottom: 15px; }
.pack-type-badge { background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); color: white; padding: 6px 14px; border-radius: 20px; font-size: 12px; font-weight: 600; text-transform: uppercase; letter-spacing: 0.5px; }
.pack-badges { display: flex; gap: 8px; margin-left: auto; }
.badge { padding: 4px 10px; border-radius: 12px; font-size: 11px; font-weight: 600; }
.badge.version { background: #e2e8f0; color: #4a5568; }
.badge.files { background: #fed7d7; color: #c53030; }
.badge.collabs { background: #c6f6d5; color: #22543d; }
.badge.popular { background: #fed7d7; color: #c53030; }
.pack-name { font-size: 20px; font-weight: 700; margin: 12px 0; color: #2d3748; }
.pack-desc { color: #718096; font-size: 14px; line-height: 1.5; margin-bottom: 15px; max-height: 60px; overflow: hidden; }
.pack-author { display: flex; align-items: center; gap: 8px; color: #718096; font-size: 13px; margin-bottom: 15px; }
.pack-stats { display: grid; grid-template-columns: repeat(3, 1fr); gap: 15px; margin: 15px 0; }
.pack-stat { text-align: center; }
.stat-icon { font-size: 16px; margin-right: 5px; }
.stat-number { font-weight: 600; color: #2d3748; }
.stat-label { font-size: 12px; color: #718096; margin-top: 2px; }
.pack-meta { display: flex; justify-content: space-between; align-items: center; margin-top: 15px; padding-top: 15px; border-top: 1px solid #e2e8f0; }
.pack-date { color: #a0aec0; font-size: 12px; }
.pack-languages { display: flex; gap: 6px; }
```

```

.lang-tag { padding: 3px 8px; background: #edf2f7; color: #4a5568; border-radius: 10px;
font-size: 11px; }
.empty-state { text-align: center; padding: 80px 20px; color: #718096; }
.empty-state h3 { color: #4a5568; margin-bottom: 10px; }
.load-more { text-align: center; margin: 50px 0; }
.load-btn { padding: 14px 50px; background: white; border: 2px solid #667eea; color:
#667eea; border-radius: 8px; cursor: pointer; font-size: 16px; font-weight: 600; transition: all
0.2s; }
.load-btn:hover { background: #667eea; color: white; }
.loading { text-align: center; padding: 40px; color: #667eea; font-size: 14px; }
.keywords { display: flex; flex-wrap: wrap; gap: 6px; margin: 12px 0; }
.keyword { padding: 4px 10px; background: #edf2f7; color: #4a5568; border-radius: 12px;
font-size: 11px; }
@media (max-width: 768px) {
  .container { padding: 15px; }
  .packs-grid { grid-template-columns: 1fr; }
  .filter-row { flex-direction: column; }
}
</style>
</head>
<body>
  <div class="container">
    <header>
      <h1 style="margin: 0; color: #2d3748;">Explore Packs</h1>
      <p style="color: #718096; margin-top: 8px;">Discover, search, and use packages created by
the community</p>

      <div class="search-box">
        <input type="text" id="searchInput" placeholder="Search packages by name, description,
keywords, or author...">
        <button onclick="searchPacks()">Search</button>
      </div>

      <button class="toggle-advanced" onclick="toggleAdvancedSearch()">
        🔍 Advanced Search Options
      </button>

      <div id="advancedSearch" class="advanced-search">
        <div class="filter-row">
          <div class="filter-group">
            <label>Package Type</label>
            <select class="filter-select" id="packageType">
              <option value="">All Types</option>
              <option value="basic">Basic</option>

```

```
        <option value="standard">Standard</option>
        <option value="advanced">Advanced</option>
    </select>
</div>
<div class="filter-group">
    <label>Language</label>
    <select class="filter-select" id="language">
        <option value="">All Languages</option>
        <option value="javascript">JavaScript/TypeScript</option>
        <option value="python">Python</option>
        <option value="wasm">WebAssembly</option>
        <option value="mixed">Mixed</option>
    </select>
</div>
<div class="filter-group">
    <label>Sort By</label>
    <select class="filter-select" id="sortBy">
        <option value="relevance">Relevance</option>
        <option value="popular">Most Popular</option>
        <option value="newest">Newest</option>
        <option value="updated">Recently Updated</option>
        <option value="name">Name (A-Z)</option>
    </select>
</div>
</div>
<div class="filter-row">
    <div class="filter-group">
        <label>Minimum Downloads</label>
        <input type="number" class="filter-input" id="minDownloads" placeholder="e.g., 100"
min="0">
    </div>
    <div class="filter-group">
        <label>Minimum Views</label>
        <input type="number" class="filter-input" id="minViews" placeholder="e.g., 1000"
min="0">
    </div>
    <div class="filter-group">
        <label>
            <input type="checkbox" id="verifiedOnly" style="margin-right: 8px;">
            Verified Packages Only
        </label>
        <label style="margin-top: 10px; display: block;">
            <input type="checkbox" id="hasReadme" style="margin-right: 8px;">
            Has README
        </label>
    </div>
</div>
```

```

        </label>
    </div>
</div>
    <button onclick="applyAdvancedFilters()" style="background: #667eea; color: white;
border: none; padding: 10px 24px; border-radius: 6px; cursor: pointer; margin-top: 10px;">
    Apply Filters
</button>
    <button onclick="resetFilters()" style="background: none; color: #667eea; border: 1px solid
#667eea; padding: 10px 24px; border-radius: 6px; cursor: pointer; margin-top: 10px; margin-left:
10px;">
    Reset
</button>
</div>

<div class="filters">
    <div class="filter-btn active" data-type="all" onclick="setFilter('all')"> ✨ All Packs</div>
    <div class="filter-btn" data-type="basic" onclick="setFilter('basic')"> 🟢 Basic</div>
    <div class="filter-btn" data-type="standard" onclick="setFilter('standard')"> 🟡
Standard</div>
    <div class="filter-btn" data-type="advanced" onclick="setFilter('advanced')"> 🟣
Advanced</div>
    <div class="filter-btn" data-type="popular" onclick="setFilter('popular')"> 🔥 Popular</div>
    <div class="filter-btn" data-type="verified" onclick="setFilter('verified')"> ✓ Verified</div>
</div>
</header>

<div id="statsBanner" class="stats-banner" style="display: none;">
    <div class="stat-item">
        <div class="stat-value" id="totalPacks">0</div>
        <div class="stat-label">Total Packs</div>
    </div>
    <div class="stat-item">
        <div class="stat-value" id="totalDownloads">0</div>
        <div class="stat-label">Total Downloads</div>
    </div>
    <div class="stat-item">
        <div class="stat-value" id="totalViews">0</div>
        <div class="stat-label">Total Views</div>
    </div>
    <div class="stat-item">
        <div class="stat-value" id="avgFiles">0</div>
        <div class="stat-label">Avg Files/Pack</div>
    </div>
</div>

```

```
<div id="packsContainer" class="packs-grid">
  <!-- Packs will be loaded here -->
</div>
```

```
<div id="loadingIndicator" class="loading" style="display: none;">
  Loading packs...
</div>
```

```
<div id="emptyState" class="empty-state" style="display: none;">
  <h3>No packs found</h3>
  <p>Try adjusting your search criteria or create your own pack!</p>
  <a href="editor.html" style="color: #667eea; text-decoration: none; font-weight: 600;">👉
Create a Pack →</a>
</div>
```

```
<div class="load-more">
  <button class="load-btn" onclick="loadMore()" id="loadMoreBtn" style="display:
none;">Load More</button>
</div>
</div>
```

```
<script>
  let currentPage = 1;
  let currentFilter = 'all';
  let currentSearch = "";
  let hasMore = false;
  let isLoading = false;
  let totalPacks = 0;
  let totalDownloads = 0;
  let totalViews = 0;
```

```
// Advanced filter state
let advancedFilters = {
  packageType: "",
  language: "",
  sort: 'relevance',
  minDownloads: 0,
  minViews: 0,
  verifiedOnly: false,
  hasReadme: false
};
```

```
// Load packs on page load
```

```
window.onload = loadPacks;

// Enter key to search
document.getElementById('searchInput').addEventListener('keypress', (e) => {
  if (e.key === 'Enter') searchPacks();
});

async function loadPacks(reset = false) {
  if (isLoading) return;

  if (reset) {
    currentPage = 1;
    document.getElementById('packsContainer').innerHTML = "";
    document.getElementById('loadMoreBtn').style.display = 'none';
    document.getElementById('emptyState').style.display = 'none';
    document.getElementById('loadingIndicator').style.display = 'block';
  }

  isLoading = true;

  try {
    const params = new URLSearchParams({
      page: currentPage,
      limit: 12,
      advanced: 'true',
      includeMetadata: 'true'
    });

    // Apply current filter
    if (currentFilter !== 'all') {
      switch(currentFilter) {
        case 'popular':
          params.set('sort', 'popular');
          params.set('minDownloads', '10');
          break;
        case 'verified':
          params.set('verified', 'true');
          break;
        default:
          params.set('type', currentFilter);
      }
    }

    // Apply search query
```

```

    if (currentSearch) params.set('q', currentSearch);

    // Apply advanced filters
    if (advancedFilters.packageType) params.set('type', advancedFilters.packageType);
    if (advancedFilters.language) params.set('language', advancedFilters.language);
    if (advancedFilters.sort) params.set('sort', advancedFilters.sort);
    if (advancedFilters.minDownloads > 0) params.set('minDownloads',
advancedFilters.minDownloads);
    if (advancedFilters.minViews > 0) params.set('minViews', advancedFilters.minViews);
    if (advancedFilters.verifiedOnly) params.set('verified', 'true');
    if (advancedFilters.hasReadme) params.set('hasReadme', 'true');

    const response = await fetch(`/api/search?${params}`);
    const result = await response.json();

    if (result.success) {
        displayPacks(result.packs);
        hasMore = result.pagination?.hasNextPage || result.hasMore;

        if (hasMore) {
            document.getElementById('loadMoreBtn').style.display = 'block';
        }

        // Update stats banner
        updateStats(result.packs, result.pagination?.total);

        // Show/hide empty state
        if (result.packs.length === 0 && currentPage === 1) {
            document.getElementById('emptyState').style.display = 'block';
            document.getElementById('statsBanner').style.display = 'none';
        } else {
            document.getElementById('emptyState').style.display = 'none';
            document.getElementById('statsBanner').style.display = 'flex';
        }
    }
} catch (error) {
    console.error('Error loading packs:', error);
    document.getElementById('packsContainer').innerHTML = `
        <div style="text-align: center; grid-column: 1/-1; color: #e53e3e; padding: 40px;">
            <h3>Error loading packs</h3>
            <p>Please try again later</p>
        </div>
    `;
} finally {

```

```

    isLoading = false;
    document.getElementById('loadingIndicator').style.display = 'none';
  }
}

function displayPacks(packs) {
  const container = document.getElementById('packsContainer');

  packs.forEach(pack => {
    const card = document.createElement('div');
    card.className = 'pack-card';
    if (pack.verificationStatus === 'verified') {
      card.classList.add('verified');
    }

    card.onclick = () => window.open(pack.workerUrl ||
`https://packcdn.firefly-worker.workers.dev/pack/${pack.urlId}`, '_blank');

    // Prepare badges
    const badges = [];
    if (pack.versions && pack.versions.length > 0) {
      badges.push(`<span class="badge version">v${pack.latestVersion ||
pack.version}</span>`);
    }
    if (pack.fileCount) {
      badges.push(`<span class="badge files">${pack.fileCount} files</span>`);
    }
    if (pack.collaboratorCount > 1) {
      badges.push(`<span class="badge collabs">${pack.collaboratorCount} collabs</span>`);
    }
    if (pack.downloads > 1000) {
      badges.push(`<span class="badge popular">Popular</span>`);
    }

    // Prepare keywords
    const keywordsHtml = pack.keywords && pack.keywords.length > 0
      ? `<div class="keywords">${pack.keywords.slice(0, 5).map(kw =>
        `<span class="keyword">${kw}</span>`).join("")}</div>`
      : "";

    // Prepare languages
    const languagesHtml = pack.languages && pack.languages.length > 0
      ? `<div class="pack-languages">${pack.languages.map(lang =>
        `<span class="lang-tag">${lang}</span>`).join("")}</div>`

```

```

: ";

card.innerHTML = `
  <div class="pack-header">
    <span class="pack-type-badge">${pack.packageType || 'basic'}</span>
    <div class="pack-badges">
      ${badges.join(",")}
    </div>
  </div>
  <div class="pack-name">${escapeHtml(pack.name)}</div>
  ${pack.description ? `<div class="pack-desc"
title="${escapeHtml(pack.description)}">${truncateText(escapeHtml(pack.description),
100)}</div>` : ""}
  ${keywordsHtml}
  <div class="pack-author">
     ${pack.author || 'Anonymous'}
    ${pack.publisherId ? `<span style="color: #a0aec0;">@${pack.publisherId}</span>` : ""}
  </div>
  <div class="pack-stats">
    <div class="pack-stat">
      <div class="stat-number">${pack.views || 0}</div>
      <div class="stat-label">Views</div>
    </div>
    <div class="pack-stat">
      <div class="stat-number">${pack.downloads || 0}</div>
      <div class="stat-label">Downloads</div>
    </div>
    <div class="pack-stat">
      <div class="stat-number">${pack.versionCount || 1}</div>
      <div class="stat-label">Versions</div>
    </div>
  </div>
  <div class="pack-meta">
    <div class="pack-date">
      ${formatDate(pack.createdAt)}
      ${pack.updatedAt !== pack.createdAt ? ` • Updated ${formatDate(pack.updatedAt,
true)}` : ""}
    </div>
    ${languagesHtml}
  </div>
`;

container.appendChild(card);
});

```

```

}

function searchPacks() {
  currentSearch = document.getElementById('searchInput').value.trim();
  loadPacks(true);
}

function setFilter(type) {
  currentFilter = type;
  document.querySelectorAll('.filter-btn').forEach(btn => {
    btn.classList.toggle('active', btn.dataset.type === type);
  });

  // Reset advanced filters when using quick filters
  if (type !== 'all') {
    resetAdvancedFilters();
  }

  loadPacks(true);
}

function loadMore() {
  if (hasMore && !isLoading) {
    currentPage++;
    loadPacks(false);
  }
}

function toggleAdvancedSearch() {
  const advancedDiv = document.getElementById('advancedSearch');
  advancedDiv.classList.toggle('active');
}

function applyAdvancedFilters() {
  advancedFilters = {
    packageType: document.getElementById('packageType').value,
    language: document.getElementById('language').value,
    sort: document.getElementById('sortBy').value,
    minDownloads: parseInt(document.getElementById('minDownloads').value) || 0,
    minViews: parseInt(document.getElementById('minViews').value) || 0,
    verifiedOnly: document.getElementById('verifiedOnly').checked,
    hasReadme: document.getElementById('hasReadme').checked
  };
}

```

```

// Reset quick filter when using advanced
currentFilter = 'all';
document.querySelectorAll('.filter-btn').forEach(btn => {
  btn.classList.toggle('active', btn.dataset.type === 'all');
});

loadPacks(true);
}

function resetFilters() {
  // Reset advanced filter inputs
  document.getElementById('packageType').value = "";
  document.getElementById('language').value = "";
  document.getElementById('sortBy').value = 'relevance';
  document.getElementById('minDownloads').value = "";
  document.getElementById('minViews').value = "";
  document.getElementById('verifiedOnly').checked = false;
  document.getElementById('hasReadme').checked = false;

  resetAdvancedFilters();
  setFilter('all');
}

function resetAdvancedFilters() {
  advancedFilters = {
    packageType: "",
    language: "",
    sort: 'relevance',
    minDownloads: 0,
    minViews: 0,
    verifiedOnly: false,
    hasReadme: false
  };
}

function updateStats(packs, total) {
  if (packs.length === 0) return;

  totalPacks = total || packs.length;
  totalDownloads = packs.reduce((sum, pack) => sum + (pack.downloads || 0), 0);
  totalViews = packs.reduce((sum, pack) => sum + (pack.views || 0), 0);
  const avgFiles = packs.reduce((sum, pack) => sum + (pack.fileCount || 1), 0) / packs.length;

  document.getElementById('totalPacks').textContent = totalPacks.toLocaleString();

```

```
document.getElementById('totalDownloads').textContent = totalDownloads.toLocaleString();
document.getElementById('totalViews').textContent = totalViews.toLocaleString();
document.getElementById('avgFiles').textContent = avgFiles.toFixed(1);
}
```

```
function formatDate(dateString, short = false) {
  if (!dateString) return '';
```

```
  const date = new Date(dateString);
  const now = new Date();
  const diff = now - date;
```

```
  const minutes = Math.floor(diff / 60000);
  const hours = Math.floor(diff / 3600000);
  const days = Math.floor(diff / 86400000);
```

```
  if (short) {
    if (minutes < 60) return `${minutes}m`;
    if (hours < 24) return `${hours}h`;
    if (days < 7) return `${days}d`;
    return date.toLocaleDateString('en-US', { month: 'short', day: 'numeric' });
  }
```

```
  if (minutes < 60) return `${minutes} minute${minutes !== 1 ? 's' : ''} ago`;
  if (hours < 24) return `${hours} hour${hours !== 1 ? 's' : ''} ago`;
  if (days < 7) return `${days} day${days !== 1 ? 's' : ''} ago`;
  return date.toLocaleDateString('en-US', { year: 'numeric', month: 'short', day: 'numeric' });
}
```

```
function truncateText(text, maxLength) {
  if (text.length <= maxLength) return text;
  return text.substr(0, maxLength) + '...';
}
```

```
function escapeHtml(text) {
  const div = document.createElement('div');
  div.textContent = text;
  return div.innerHTML;
}
```

```
// Infinite scroll
```

```
window.addEventListener('scroll', () => {
  if (window.innerHeight + window.scrollY >= document.body.offsetHeight - 100) {
    loadMore();
  }
});
```

```
    }  
  });  
</script>  
</body>  
</html>
```

public/Login/login.html

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>PackCDN - Secure Anonymous Login</title>  
  <style>  
    * {  
      margin: 0;  
      padding: 0;  
      box-sizing: border-box;  
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
    }  
  
    body {  
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
      min-height: 100vh;  
      display: flex;  
      align-items: center;  
      justify-content: center;  
      padding: 20px;  
    }  
  
    .login-container {  
      background: rgba(255, 255, 255, 0.95);  
      backdrop-filter: blur(10px);  
      border-radius: 20px;  
      box-shadow: 0 20px 60px rgba(0, 0, 0, 0.3);  
      width: 100%;  
      max-width: 450px;  
      overflow: hidden;  
      border: 1px solid rgba(255, 255, 255, 0.2);  
    }  
  
    .login-header {  
      background: linear-gradient(135deg, #2563eb 0%, #1d4ed8 100%);  
      padding: 30px;
```

```
        text-align: center;
        color: white;
    }

    .login-header h1 {
        font-size: 28px;
        font-weight: 600;
        margin-bottom: 5px;
    }

    .login-header p {
        opacity: 0.9;
        font-size: 14px;
    }

    .login-body {
        padding: 40px;
    }

    .step {
        display: none;
        animation: fadeIn 0.3s ease;
    }

    .step.active {
        display: block;
    }

    @keyframes fadeIn {
        from { opacity: 0; transform: translateY(10px); }
        to { opacity: 1; transform: translateY(0); }
    }

    .form-group {
        margin-bottom: 25px;
    }

    label {
        display: block;
        margin-bottom: 8px;
        color: #374151;
        font-weight: 500;
        font-size: 14px;
    }
```

```
input {
  width: 100%;
  padding: 14px 16px;
  border: 2px solid #e5e7eb;
  border-radius: 10px;
  font-size: 16px;
  transition: all 0.3s;
  background: white;
}

input:focus {
  outline: none;
  border-color: #2563eb;
  box-shadow: 0 0 0 3px rgba(37, 99, 235, 0.1);
}

.username-display {
  background: #f3f4f6;
  padding: 14px 16px;
  border-radius: 10px;
  border: 2px solid #e5e7eb;
  font-family: monospace;
  font-size: 18px;
  font-weight: 600;
  color: #1f2937;
  text-align: center;
  letter-spacing: 1px;
}

.checkbox-group {
  display: flex;
  align-items: center;
  gap: 10px;
  margin: 25px 0;
}

.checkbox-group input[type="checkbox"] {
  width: auto;
  transform: scale(1.2);
}

.btn {
  width: 100%;
```

```
padding: 16px;
border: none;
border-radius: 10px;
font-size: 16px;
font-weight: 600;
cursor: pointer;
transition: all 0.3s;
display: flex;
align-items: center;
justify-content: center;
gap: 10px;
}

.btn-primary {
  background: linear-gradient(135deg, #2563eb 0%, #1d4ed8 100%);
  color: white;
}

.btn-primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 10px 25px rgba(37, 99, 235, 0.3);
}

.btn-secondary {
  background: #6b7280;
  color: white;
}

.btn-secondary:hover {
  background: #4b5563;
}

.loading {
  display: none;
  text-align: center;
  padding: 20px;
}

.loading.active {
  display: block;
}

.spinner {
  width: 40px;
```

```
height: 40px;
border: 3px solid #e5e7eb;
border-top-color: #2563eb;
border-radius: 50%;
animation: spin 1s linear infinite;
margin: 0 auto 15px;
}
```

```
@keyframes spin {
  to { transform: rotate(360deg); }
}
```

```
.message {
  padding: 12px 16px;
  border-radius: 8px;
  margin-bottom: 20px;
  font-size: 14px;
  display: none;
}
```

```
.message.success {
  background: #d1fae5;
  color: #065f46;
  border: 1px solid #a7f3d0;
  display: block;
}
```

```
.message.error {
  background: #fee2e2;
  color: #991b1b;
  border: 1px solid #fecaca;
  display: block;
}
```

```
.message.info {
  background: #dbeafe;
  color: #1e40af;
  border: 1px solid #bfdbfe;
  display: block;
}
```

```
.step-indicator {
  display: flex;
  justify-content: center;
```

```
    gap: 10px;
    margin-bottom: 30px;
}
```

```
.step-dot {
    width: 10px;
    height: 10px;
    border-radius: 50%;
    background: #e5e7eb;
    transition: all 0.3s;
}
```

```
.step-dot.active {
    background: #2563eb;
    transform: scale(1.2);
}
```

```
.codes-info {
    background: #fef3c7;
    border: 1px solid #fbbf24;
    padding: 15px;
    border-radius: 10px;
    margin-bottom: 25px;
    font-size: 13px;
    color: #92400e;
}
```

```
.codes-info strong {
    color: #d97706;
}
```

```
.back-btn {
    background: none;
    border: none;
    color: #6b7280;
    cursor: pointer;
    font-size: 14px;
    display: flex;
    align-items: center;
    gap: 5px;
    margin-bottom: 20px;
}
```

```
.back-btn:hover {
```

```

        color: #374151;
    }

    .security-badge {
        display: inline-flex;
        align-items: center;
        gap: 8px;
        background: #dcfce7;
        color: #166534;
        padding: 6px 12px;
        border-radius: 20px;
        font-size: 12px;
        font-weight: 500;
        margin-top: 10px;
    }
</style>
</head>
<body>
    <div class="login-container">
        <div class="login-header">
            <h1>PackCDN</h1>
            <p>Secure Anonymous Access</p>
            <div class="security-badge">
                <svg width="16" height="16" viewBox="0 0 24 24" fill="currentColor">
                    <path d="M12 1L3 5v6c0 5.55 3.84 10.74 9 12 5.16-1.26 9-6.45 9-12V5l-9-4z"/>
                </svg>
                HTTPS-Only • No Passwords • Anonymous
            </div>
        </div>
        <div class="login-body">
            <div class="step-indicator">
                <div class="step-dot active" data-step="1"></div>
                <div class="step-dot" data-step="2"></div>
            </div>

            <!-- Step 1: Email Entry -->
            <div class="step active" id="step1">
                <h2 style="margin-bottom: 20px; color: #1f2937;">Get Anonymous Access</h2>
                <p style="color: #6b7280; margin-bottom: 25px; line-height: 1.6;">
                    Enter your email to receive 20 secure recovery codes. Each code can be used only
                    once for login.
                </p>
            </div>
        </div>
    </div>

```

```

<div class="message" id="message1"></div>

<div class="form-group">
  <label for="email">Your Email Address</label>
  <input type="email" id="email" placeholder="you@example.com" required>
</div>

<div class="checkbox-group">
  <input type="checkbox" id="terms">
  <label for="terms">I understand that I must save my recovery codes
securely</label>
</div>

<button class="btn btn-primary" onclick="initiateLogin()" id="initiateBtn">
  <span>Send Recovery Codes</span>
  <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2">
    <path d="M22 2L11 13M22 2L-7 20-4-9-9-4 20-7z"/>
  </svg>
</button>
</div>

<!-- Step 2: Code Verification -->
<div class="step" id="step2">
  <button class="back-btn" onclick="goBack()">
    <svg width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2">
      <path d="M19 12H5M12 19L-7 7 7-7"/>
    </svg>
    Back
  </button>

  <h2 style="margin-bottom: 20px; color: #1f2937;">Enter Recovery Code</h2>

  <div class="username-display" id="generatedUsername"></div>
  <p style="color: #6b7280; margin: 10px 0 25px; font-size: 13px; text-align: center;">
    This is your anonymous username
  </p>

  <div class="message" id="message2"></div>

  <div class="codes-info">
    <strong>📧 Check your email</strong> for the list of 20 recovery codes.
    Use them in order (Code #1 first, then #2, etc.).
  </div>

```

```

</div>

<div class="form-group">
  <label for="code">Recovery Code</label>
  <input type="text" id="code" placeholder="Enter code (e.g., A1B2C3)" required>
</div>

<div class="checkbox-group">
  <input type="checkbox" id="rememberMe">
  <label for="rememberMe">Remember me for 30 days (secure HTTPS
cookie)</label>
</div>

<button class="btn btn-primary" onclick="verifyCode()" id="verifyBtn">
  <span>Verify & Login</span>
  <svg width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2">
    <path d="M5 13l4 4L19 7"/>
  </svg>
</button>
</div>

<!-- Loading State -->
<div class="loading" id="loading">
  <div class="spinner"></div>
  <p id="loadingText">Processing...</p>
</div>
</div>
</div>

<script>
  let currentStep = 1;
  let generatedUsername = "";

  function showStep(step) {
    document.querySelectorAll('.step').forEach(s => s.classList.remove('active'));
    document.querySelectorAll('.step-dot').forEach(d => d.classList.remove('active'));

    document.getElementById(`step${step}`).classList.add('active');
    document.querySelector(`.step-dot[data-step="${step}"]`).classList.add('active');
    currentStep = step;
  }

  function showMessage(elementId, message, type = 'info') {

```

```
const element = document.getElementById(elementId);
element.textContent = message;
element.className = `message ${type}`;
element.style.display = 'block';
}

function hideMessage(elementId) {
  document.getElementById(elementId).style.display = 'none';
}

function setLoading/loading, text = 'Processing...') {
  const loadingEl = document.getElementById('loading');
  const loadingText = document.getElementById('loadingText');

  loadingText.textContent = text;

  if (loading) {
    loadingEl.classList.add('active');
    document.getElementById('initiateBtn').disabled = true;
    document.getElementById('verifyBtn').disabled = true;
  } else {
    loadingEl.classList.remove('active');
    document.getElementById('initiateBtn').disabled = false;
    document.getElementById('verifyBtn').disabled = false;
  }
}

async function initiateLogin() {
  const email = document.getElementById('email').value.trim();
  const terms = document.getElementById('terms').checked;

  if (!email) {
    showMessage('message1', 'Please enter your email address', 'error');
    return;
  }

  if (!terms) {
    showMessage('message1', 'You must agree to securely save your recovery codes',
'error');
    return;
  }

  setLoading(true, 'Generating secure credentials...');
```

```

try {
  const response = await fetch('/api/login', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      action: 'initiate',
      email: email
    })
  });

  const data = await response.json();

  if (response.ok) {
    generatedUsername = data.username;
    document.getElementById('generatedUsername').textContent = data.username;

    showMessage('message1',
      '✅ Recovery codes sent to your email! Check your inbox (and spam folder).',
      'success'
    );

    setTimeout(() => {
      hideMessage('message1');
      showStep(2);
    }, 2000);
  } else {
    showMessage('message1', data.error || 'Failed to send recovery codes', 'error');
  }
} catch (error) {
  showMessage('message1', 'Network error. Please try again.', 'error');
} finally {
  setLoading(false);
}
}

async function verifyCode() {
  const email = document.getElementById('email').value.trim();
  const code = document.getElementById('code').value.trim().toUpperCase();
  const rememberMe = document.getElementById('rememberMe').checked;

  if (!code) {
    showMessage('message2', 'Please enter your recovery code', 'error');
  }
}

```

```

    return;
  }

  setLoading(true, 'Verifying recovery code...');

  try {
    const response = await fetch('/api/login', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        action: 'verify',
        email: email,
        code: code,
        rememberMe: rememberMe
      })
    });

    const data = await response.json();

    if (response.ok) {
      showMessage('message2',
        `✅ Login successful! Welcome ${data.username}. ${data.codesRemaining}
codes remaining.`,
        'success'
      );

      // Redirect to dashboard after successful login
      setTimeout(() => {
        window.location.href = '/dashboard';
      }, 1500);
    } else {
      showMessage('message2', data.error || 'Invalid recovery code', 'error');
    }
  } catch (error) {
    showMessage('message2', 'Network error. Please try again.', 'error');
  } finally {
    setLoading(false);
  }
}

function goBack() {
  showStep(1);
}

```

```
}

// Enter key support
document.getElementById('email').addEventListener('keypress', function(e) {
    if (e.key === 'Enter') initiateLogin();
});

document.getElementById('code').addEventListener('keypress', function(e) {
    if (e.key === 'Enter') verifyCode();
});
</script>
</body>
</html>
```

public/Private/admin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>QUANTUM FIREWALL ADMIN | SECURE DATABASE MANAGEMENT</title>
    <style>
        :root {
            --firewall-red: #ff0033;
            --firewall-orange: #ff6600;
            --firewall-yellow: #ffcc00;
            --firewall-green: #00ff88;
            --firewall-blue: #0066ff;
            --firewall-purple: #9900ff;
            --matrix-green: #00ff41;
            --terminal-bg: #0a0a0a;
            --panel-bg: #111111;
            --panel-border: #222222;
            --text-primary: #ffffff;
            --text-secondary: #888888;
            --danger: #ff3333;
            --warning: #ffaa00;
            --success: #00cc66;
            --info: #0099ff;
        }

        * {
            margin: 0;
            padding: 0;
```

```
    box-sizing: border-box;
    font-family: 'Courier New', 'JetBrains Mono', 'Fira Code', monospace;
}

body {
    background: var(--terminal-bg);
    color: var(--matrix-green);
    min-height: 100vh;
    overflow-x: hidden;
    background-image:
        radial-gradient(circle at 10% 20%, rgba(0, 255, 65, 0.05) 0%, transparent 20%),
        radial-gradient(circle at 90% 80%, rgba(255, 0, 51, 0.05) 0%, transparent 20%);
}

/* Matrix Background Effect */
.matrix-bg {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: -1;
    opacity: 0.1;
    pointer-events: none;
}

/* Main Container */
.firewall-container {
    max-width: 1400px;
    margin: 0 auto;
    padding: 20px;
}

/* Header */
.firewall-header {
    background: linear-gradient(90deg, rgba(255,0,51,0.1), rgba(0,255,65,0.1));
    border: 2px solid var(--firewall-red);
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 30px;
    position: relative;
    overflow: hidden;
    box-shadow: 0 0 30px rgba(255, 0, 51, 0.3);
}
```

```
.firewall-header::before {  
  content: "";  
  position: absolute;  
  top: 0;  
  left: -100%;  
  width: 100%;  
  height: 100%;  
  background: linear-gradient(90deg, transparent, rgba(255,0,51,0.2), transparent);  
  animation: scanning 3s infinite;  
}
```

```
@keyframes scanning {  
  0% { left: -100%; }  
  100% { left: 100%; }  
}
```

```
.header-title {  
  display: flex;  
  align-items: center;  
  gap: 15px;  
  margin-bottom: 10px;  
}
```

```
.header-title h1 {  
  font-size: 28px;  
  font-weight: bold;  
  text-shadow: 0 0 10px var(--firewall-red);  
  letter-spacing: 2px;  
}
```

```
.security-level {  
  display: inline-block;  
  padding: 5px 15px;  
  background: var(--danger);  
  color: white;  
  border-radius: 4px;  
  font-size: 12px;  
  font-weight: bold;  
  animation: pulse 2s infinite;  
}
```

```
@keyframes pulse {  
  0%, 100% { opacity: 1; }
```

```
    50% { opacity: 0.7; }
}

.status-bar {
  display: flex;
  gap: 20px;
  margin-top: 15px;
}

.status-item {
  display: flex;
  align-items: center;
  gap: 8px;
}

.status-dot {
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background: var(--success);
}

.status-dot.warning { background: var(--warning); }
.status-dot.danger { background: var(--danger); }
.status-dot.success { background: var(--success); }

/* Login Panel */
.login-panel {
  background: var(--panel-bg);
  border: 1px solid var(--panel-border);
  border-radius: 8px;
  padding: 30px;
  margin-bottom: 30px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
}

.panel-title {
  display: flex;
  align-items: center;
  gap: 10px;
  margin-bottom: 20px;
  padding-bottom: 10px;
  border-bottom: 1px solid var(--panel-border);
}
```

```
.panel-title h2 {
  font-size: 18px;
  color: var(--firewall-green);
}

.password-input-container {
  position: relative;
  margin-bottom: 20px;
}

.password-input {
  width: 100%;
  padding: 15px;
  background: rgba(0, 0, 0, 0.5);
  border: 2px solid var(--firewall-blue);
  border-radius: 4px;
  color: var(--matrix-green);
  font-size: 16px;
  letter-spacing: 2px;
  transition: all 0.3s;
}

.password-input:focus {
  outline: none;
  border-color: var(--firewall-green);
  box-shadow: 0 0 15px rgba(0, 255, 65, 0.3);
}

.password-strength {
  height: 4px;
  background: linear-gradient(90deg, var(--danger), var(--warning), var(--success));
  margin-top: 5px;
  border-radius: 2px;
  overflow: hidden;
}

.strength-meter {
  height: 100%;
  width: 0%;
  background: var(--success);
  transition: width 0.3s;
}
```

```
.access-buttons {
  display: flex;
  gap: 15px;
  margin-top: 25px;
}

.btn {
  padding: 12px 24px;
  border: none;
  border-radius: 4px;
  font-size: 14px;
  font-weight: bold;
  cursor: pointer;
  transition: all 0.3s;
  display: flex;
  align-items: center;
  gap: 8px;
}

.btn-primary {
  background: linear-gradient(135deg, var(--firewall-blue), var(--firewall-purple));
  color: white;
  flex: 1;
}

.btn-primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 5px 20px rgba(0, 102, 255, 0.4);
}

.btn-secondary {
  background: transparent;
  border: 2px solid var(--firewall-red);
  color: var(--firewall-red);
}

.btn-secondary:hover {
  background: var(--firewall-red);
  color: white;
}

/* Firewall Logs */
.firewall-logs {
  background: var(--panel-bg);
```

```
border: 1px solid var(--panel-border);
border-radius: 8px;
padding: 20px;
margin-bottom: 30px;
height: 300px;
overflow-y: auto;
}

.log-entry {
padding: 10px;
margin-bottom: 8px;
border-left: 3px solid var(--info);
background: rgba(0, 0, 0, 0.3);
font-size: 12px;
animation: slideIn 0.3s ease-out;
}

@keyframes slideIn {
from { transform: translateX(-10px); opacity: 0; }
to { transform: translateX(0); opacity: 1; }
}

.log-entry.warning { border-left-color: var(--warning); }
.log-entry.danger { border-left-color: var(--danger); }
.log-entry.success { border-left-color: var(--success); }

.log-timestamp {
color: var(--text-secondary);
font-size: 11px;
margin-right: 10px;
}

/* Admin Interface (Hidden until login) */
#adminInterface {
display: none;
}

.admin-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
gap: 20px;
margin-bottom: 30px;
}
```

```
.admin-panel {
  background: var(--panel-bg);
  border: 1px solid var(--panel-border);
  border-radius: 8px;
  padding: 20px;
  height: 100%;
}

.admin-panel h3 {
  color: var(--firewall-blue);
  margin-bottom: 15px;
  font-size: 16px;
  display: flex;
  align-items: center;
  gap: 8px;
}

.table-list {
  max-height: 200px;
  overflow-y: auto;
}

.table-item {
  padding: 10px;
  margin-bottom: 5px;
  background: rgba(0, 0, 0, 0.3);
  border-radius: 4px;
  cursor: pointer;
  transition: all 0.2s;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.table-item:hover {
  background: rgba(0, 102, 255, 0.2);
  border-left: 3px solid var(--firewall-blue);
}

.table-item.active {
  background: rgba(0, 102, 255, 0.3);
  border-left: 3px solid var(--firewall-green);
}
```

```
.row-count {
    background: rgba(0, 255, 65, 0.1);
    color: var(--firewall-green);
    padding: 2px 8px;
    border-radius: 10px;
    font-size: 11px;
}

/* SQL Editor */
.sql-editor {
    width: 100%;
    height: 200px;
    background: #1a1a1a;
    border: 1px solid var(--panel-border);
    border-radius: 4px;
    padding: 15px;
    color: var(--matrix-green);
    font-family: 'Courier New', monospace;
    font-size: 14px;
    resize: vertical;
    margin-bottom: 10px;
}

.sql-editor:focus {
    outline: none;
    border-color: var(--firewall-green);
}

/* Data Table */
.data-table-container {
    overflow-x: auto;
    margin-top: 20px;
    max-height: 500px;
}

.data-table {
    width: 100%;
    border-collapse: collapse;
    font-size: 12px;
}

.data-table th {
    background: rgba(0, 102, 255, 0.2);
    color: var(--firewall-blue);
```

```
padding: 12px;
text-align: left;
font-size: 12px;
border-bottom: 2px solid var(--panel-border);
position: sticky;
top: 0;
}

.data-table td {
padding: 10px;
border-bottom: 1px solid var(--panel-border);
font-size: 12px;
max-width: 200px;
overflow: hidden;
text-overflow: ellipsis;
white-space: nowrap;
}

.data-table tr:hover {
background: rgba(0, 255, 65, 0.05);
}

/* Action Buttons */
.action-buttons {
display: flex;
gap: 10px;
margin-top: 20px;
flex-wrap: wrap;
}

.btn-small {
padding: 6px 12px;
font-size: 12px;
border: none;
border-radius: 4px;
cursor: pointer;
transition: all 0.2s;
display: flex;
align-items: center;
gap: 5px;
}

.btn-edit { background: var(--warning); color: black; }
.btn-delete { background: var(--danger); color: white; }
```

```
.btn-execute { background: var(--success); color: white; }  
.btn-view { background: var(--info); color: white; }  
.btn-export { background: var(--firewall-purple); color: white; }
```

```
.btn-small:hover {  
  transform: translateY(-1px);  
  opacity: 0.9;  
}
```

```
.btn-small:disabled {  
  opacity: 0.5;  
  cursor: not-allowed;  
  transform: none;  
}
```

```
/* Modal */
```

```
.modal {  
  display: none;  
  position: fixed;  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background: rgba(0, 0, 0, 0.8);  
  z-index: 1000;  
  justify-content: center;  
  align-items: center;  
}
```

```
.modal-content {  
  background: var(--panel-bg);  
  border: 2px solid var(--firewall-red);  
  border-radius: 8px;  
  padding: 30px;  
  width: 90%;  
  max-width: 500px;  
  max-height: 80vh;  
  overflow-y: auto;  
}
```

```
.modal-header {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;
```

```
    margin-bottom: 20px;
    padding-bottom: 10px;
    border-bottom: 1px solid var(--panel-border);
}
```

```
.modal-title {
    color: var(--firewall-red);
    font-size: 18px;
}
```

```
.close-modal {
    background: none;
    border: none;
    color: var(--text-secondary);
    font-size: 24px;
    cursor: pointer;
}
```

```
.form-group {
    margin-bottom: 15px;
}
```

```
.form-label {
    display: block;
    margin-bottom: 5px;
    color: var(--text-secondary);
    font-size: 12px;
}
```

```
.form-input {
    width: 100%;
    padding: 10px;
    background: rgba(0, 0, 0, 0.5);
    border: 1px solid var(--panel-border);
    border-radius: 4px;
    color: var(--matrix-green);
    font-family: 'Courier New', monospace;
}
```

```
/* JSON Viewer */
```

```
.json-viewer {
    background: #1a1a1a;
    border: 1px solid var(--panel-border);
    border-radius: 4px;
```

```
padding: 15px;
font-family: 'Courier New', monospace;
font-size: 12px;
max-height: 300px;
overflow-y: auto;
white-space: pre-wrap;
color: var(--matrix-green);
}
```

```
/* Terminal Style */
.terminal {
  background: rgba(0, 0, 0, 0.8);
  border: 1px solid var(--matrix-green);
  border-radius: 4px;
  padding: 15px;
  font-family: 'Courier New', monospace;
  font-size: 12px;
  line-height: 1.4;
  overflow-x: auto;
  min-height: 100px;
}
```

```
.terminal-line {
  margin-bottom: 5px;
  display: flex;
  gap: 10px;
}
```

```
.terminal-prompt {
  color: var(--firewall-green);
  flex-shrink: 0;
}
```

```
.terminal-command {
  color: white;
}
```

```
.terminal-output {
  color: var(--text-secondary);
  flex-grow: 1;
}
```

```
/* Animations */
.blink {
```

```

        animation: blink 1s infinite;
    }

    @keyframes blink {
        0%, 100% { opacity: 1; }
        50% { opacity: 0.5; }
    }

    .glow {
        animation: glow 2s infinite alternate;
    }

    @keyframes glow {
        from { box-shadow: 0 0 5px var(--firewall-green); }
        to { box-shadow: 0 0 20px var(--firewall-green); }
    }

    /* Footer */
    .firewall-footer {
        text-align: center;
        padding: 20px;
        color: var(--text-secondary);
        font-size: 12px;
        border-top: 1px solid var(--panel-border);
        margin-top: 30px;
    }

    /* Responsive */
    @media (max-width: 768px) {
        .admin-grid {
            grid-template-columns: 1fr;
        }

        .access-buttons {
            flex-direction: column;
        }

        .status-bar {
            flex-wrap: wrap;
        }
    }
</style>
</head>
<body>

```

```

<!-- Matrix Background -->
<div class="matrix-bg" id="matrixCanvas"></div>

<!-- Modals -->
<div class="modal" id="insertModal">
  <div class="modal-content">
    <div class="modal-header">
      <h3 class="modal-title">✚ INSERT ROW</h3>
      <button class="close-modal" onclick="closeModal('insertModal')">✕</button>
    </div>
    <div id="insertForm">
      <!-- Dynamic form will be inserted here -->
    </div>
    <div class="action-buttons">
      <button class="btn btn-execute btn-small" onclick="submitInsert()">INSERT</button>
      <button class="btn btn-secondary btn-small"
onclick="closeModal('insertModal')">CANCEL</button>
    </div>
  </div>
</div>

<div class="modal" id="updateModal">
  <div class="modal-content">
    <div class="modal-header">
      <h3 class="modal-title">✎ UPDATE ROW</h3>
      <button class="close-modal" onclick="closeModal('updateModal')">✕</button>
    </div>
    <div id="updateForm">
      <!-- Dynamic form will be inserted here -->
    </div>
    <div class="action-buttons">
      <button class="btn btn-execute btn-small"
onclick="submitUpdate()">UPDATE</button>
      <button class="btn btn-secondary btn-small"
onclick="closeModal('updateModal')">CANCEL</button>
    </div>
  </div>
</div>

<div class="modal" id="viewModal">
  <div class="modal-content">
    <div class="modal-header">
      <h3 class="modal-title">👁 VIEW DATA</h3>
      <button class="close-modal" onclick="closeModal('viewModal')">✕</button>
    </div>
  </div>
</div>

```

```

        </div>
        <div class="json-viewer" id="viewData">
            { /* JSON data will be displayed here */ }
        </div>
    </div>
</div>

<div class="modal" id="confirmModal">
    <div class="modal-content">
        <div class="modal-header">
            <h3 class="modal-title"> ⚠️ CONFIRM ACTION</h3>
            <button class="close-modal" onclick="closeModal('confirmModal')">×</button>
        </div>
        <div style="margin: 20px 0;">
            <p id="confirmMessage">Are you sure you want to perform this action?</p>
        </div>
        <div class="action-buttons">
            <button class="btn btn-delete btn-small"
onclick="confirmAction()">CONFIRM</button>
            <button class="btn btn-secondary btn-small"
onclick="closeModal('confirmModal')">CANCEL</button>
        </div>
    </div>
</div>

<!-- Main Interface -->
<div class="firewall-container">
    <!-- Header -->
    <div class="firewall-header">
        <div class="header-title">
            <h1>QUANTUM FIREWALL ADMIN v2.0</h1>
            <span class="security-level">MAXIMUM SECURITY</span>
        </div>
        <p>Secure Supabase Database Management Interface | Encrypted Connection
Required</p>

        <div class="status-bar">
            <div class="status-item">
                <span class="status-dot success"></span>
                <span>FIREWALL: ACTIVE</span>
            </div>
            <div class="status-item">
                <span class="status-dot warning blink"></span>
                <span>ENCRYPTION: 256-BIT AES</span>
            </div>
        </div>
    </div>

```

```

    </div>
    <div class="status-item">
      <span class="status-dot" id="sessionStatusDot"></span>
      <span id="sessionStatus">SESSION: NOT AUTHENTICATED</span>
    </div>
    <div class="status-item">
      <span class="status-dot danger"></span>
      <span>THREAT LEVEL: HIGH</span>
    </div>
  </div>
</div>

<!-- Login Panel -->
<div class="login-panel" id="loginPanel">
  <div class="panel-title">
    <h2>🔒 SECURE AUTHENTICATION REQUIRED</h2>
  </div>

  <div class="password-input-container">
    <input type="password"
      id="password"
      class="password-input"
      placeholder="ENTER ADMINISTRATOR PASSWORD"
      maxlength="32"
      autocomplete="off"
      onkeypress="if(event.key === 'Enter') secureLogin()">
    <div class="password-strength">
      <div class="strength-meter" id="strengthMeter"></div>
    </div>
  </div>

  <div class="terminal">
    <div class="terminal-line">
      <span class="terminal-prompt">firewall@admin:~$ </span>
      <span class="terminal-command">connect --secure --admin</span>
    </div>
    <div class="terminal-line">
      <span class="terminal-prompt">system@auth:~$ </span>
      <span class="terminal-output">Establishing quantum-encrypted tunnel...</span>
    </div>
    <div class="terminal-line">
      <span class="terminal-prompt">system@auth:~$ </span>
      <span class="terminal-output" id="authStatus">Waiting for credentials...</span>
    </div>
  </div>

```

```
</div>

<div class="access-buttons">
  <button class="btn btn-primary" onclick="secureLogin()">
    <span>🚀 INITIATE SECURE CONNECTION</span>
    <span>🔒</span>
  </button>
  <button class="btn btn-secondary" onclick="resetInterface()">
    <span>🚨 EMERGENCY LOCKDOWN</span>
    <span>⚠️</span>
  </button>
</div>
</div>

<!-- Firewall Logs -->
<div class="firewall-logs" id="firewallLogs">
  <div class="log-entry">
    <span class="log-timestamp" id="currentTime"></span>
    <span>SYSTEM: Quantum firewall initialized</span>
  </div>
  <div class="log-entry">
    <span class="log-timestamp" id="currentTime2"></span>
    <span>SECURITY: All encryption layers active</span>
  </div>
</div>

<!-- Admin Interface (Hidden until login) -->
<div id="adminInterface">
  <div class="admin-grid">
    <!-- Database Overview -->
    <div class="admin-panel">
      <h3>📊 DATABASE OVERVIEW</h3>
      <div class="table-list" id="tableList">
        <div style="text-align: center; padding: 20px; color: var(--text-secondary);">
          Click "Refresh Tables" to load database structure
        </div>
      </div>
    </div>
    <div class="action-buttons">
      <button class="btn btn-primary btn-small" onclick="getAllTables()">
        🔄 Refresh Tables
      </button>
      <button class="btn btn-secondary btn-small" onclick="clearSelection()">
        🗑️ Clear Selection
      </button>
    </div>
  </div>
</div>
```

```

    </div>
</div>

<!-- SQL Editor -->
<div class="admin-panel">
    <h3>🖥️ SQL COMMAND INTERFACE</h3>
    <textarea class="sql-editor"
        id="sqlEditor"
        placeholder="-- Enter SQL commands here\n-- Example: SELECT * FROM
users LIMIT 10;\n-- INSERT INTO table (column) VALUES ('value');\n-- UPDATE table SET
column = 'value' WHERE id = 1;"></textarea>
    <div class="action-buttons">
        <button class="btn btn-execute btn-small" onclick="executeSQL()">
            ⚡ Execute Query
        </button>
        <button class="btn btn-view btn-small" onclick="explainSQL()">
            🔍 Explain Query
        </button>
        <button class="btn btn-secondary btn-small" onclick="clearEditor()">
            ✏️ Clear
        </button>
    </div>
</div>

<!-- Data Operations -->
<div class="admin-panel">
    <h3>⚙️ DATA OPERATIONS</h3>
    <div class="terminal" id="operationTerminal">
        <div class="terminal-line">
            <span class="terminal-prompt">db@ops:~$ </span>
            <span class="terminal-output" id="operationStatus">Select a table to begin
operations</span>
        </div>
    </div>
    <div class="action-buttons">
        <button class="btn btn-edit btn-small" onclick="showInsertModal()" disabled
id="btnInsert">
            + Insert Row
        </button>
        <button class="btn btn-edit btn-small" onclick="showUpdateModal()" disabled
id="btnUpdate">
            ✏️ Update Row
        </button>
    </div>
</div>

```

```
        <button class="btn btn-delete btn-small" onclick="showDeleteConfirm()" disabled  
id="btnDelete">
```

 Delete Row

```
</button>
```

```
        <button class="btn btn-view btn-small" onclick="viewRowData()" disabled  
id="btnView">
```

 View Data

```
</button>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- Data Table Display -->
```

```
<div class="admin-panel">
```

```
    <h3>  TABLE DATA: <span id="currentTable">None Selected</span></h3>
```

```
    <div class="data-table-container">
```

```
        <table class="data-table" id="dataTable">
```

```
            <thead>
```

```
                <tr>
```

```
                    <th style="width: 50px;">#</th>
```

```
                    <th>Select</th>
```

```
                    <!-- Headers will be populated dynamically -->
```

```
                </tr>
```

```
            </thead>
```

```
            <tbody>
```

```
                <tr>
```

```
                    <td colspan="10" style="text-align: center; padding: 40px; color:  
var(--text-secondary);">
```

```
                        Select a table to view data
```

```
                    </td>
```

```
                </tr>
```

```
            </tbody>
```

```
        </table>
```

```
    </div>
```

```
    <div class="action-buttons">
```

```
        <button class="btn btn-export btn-small" onclick="exportData('csv')">
```

 Export CSV

```
</button>
```

```
        <button class="btn btn-export btn-small" onclick="exportData('json')">
```

 Export JSON

```
</button>
```

```
        <button class="btn btn-secondary btn-small" onclick="refreshData()">
```

 Refresh Data

```
</button>
```

```
<div style="margin-left: auto; display: flex; align-items: center; gap: 10px;">
  <span style="font-size: 11px; color: var(--text-secondary);">Rows:</span>
  <select id="rowLimit" style="padding: 5px; background: rgba(0,0,0,0.5); border:
1px solid var(--panel-border); color: var(--matrix-green);">
    <option value="10">10</option>
    <option value="50">50</option>
    <option value="100" selected>100</option>
    <option value="500">500</option>
    <option value="1000">1000</option>
  </select>
</div>
</div>
</div>
```

```
<!-- Connection Status -->
<div class="admin-panel">
  <h3>🔗 CONNECTION STATUS</h3>
  <div class="terminal">
    <div class="terminal-line">
      <span class="terminal-prompt">status@supabase:~$ </span>
      <span class="terminal-output" id="supabaseStatus">Connected to: <span
id="supabaseUrl">...</span></span></div>
    <div class="terminal-line">
      <span class="terminal-prompt">status@tables:~$ </span>
      <span class="terminal-output">Tables loaded: <span
id="tableCount">0</span></span></div>
    <div class="terminal-line">
      <span class="terminal-prompt">status@session:~$ </span>
      <span class="terminal-output">Session ID: <span
id="sessionId">...</span></span></div>
  </div>
</div>
</div>
</div>
```

```
<!-- Footer -->
<div class="firewall-footer">
  <p>⚠️ WARNING: This interface is protected by quantum-grade encryption. All actions
are logged and monitored.</p>
  <p>© 2024 QUANTUM FIREWALL SYSTEM | Version 2.0 | Session: <span
id="sessionTimer">00:00:00</span></p>
</div>
```

</div>

<script>

```
// =====
// GLOBAL STATE
// =====
let currentSession = {
  authenticated: false,
  token: null,
  tables: [],
  currentTable: null,
  currentData: null,
  selectedRows: new Set(),
  startTime: null,
  supabaseUrl: null
};

// =====
// INITIALIZATION
// =====
document.addEventListener('DOMContentLoaded', function() {
  updateTimestamps();
  setInterval(updateTimestamps, 1000);
  setInterval(updateSessionTimer, 1000);

  // Add initial logs
  addLog('SYSTEM: Quantum firewall initialized', 'info');
  addLog('SECURITY: All encryption layers active', 'info');
  addLog('NETWORK: Waiting for authentication...', 'warning');
});

function updateTimestamps() {
  const now = new Date();
  const timestamp = now.toISOString().replace('T', ' ').substring(0, 19);
  document.querySelectorAll('.log-timestamp').forEach(el => {
    if(el.id === 'currentTime' || el.id === 'currentTime2') {
      el.textContent = `[${timestamp}]`;
    }
  });
}

// =====
// MATRIX BACKGROUND
// =====
```

```

const matrixCanvas = document.getElementById('matrixCanvas');
const ctx = matrixCanvas.getContext('2d');
matrixCanvas.width = window.innerWidth;
matrixCanvas.height = window.innerHeight;

const chars =
"01ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789#$%^&*";
const charArray = chars.split("");
const fontSize = 14;
const columns = matrixCanvas.width / fontSize;
const drops = Array(Math.floor(columns)).fill(1);

function drawMatrix() {
  ctx.fillStyle = "rgba(0, 0, 0, 0.04)";
  ctx.fillRect(0, 0, matrixCanvas.width, matrixCanvas.height);

  ctx.fillStyle = "#00ff41";
  ctx.font = `${fontSize}px 'Courier New', monospace`;

  drops.forEach((drop, i) => {
    const char = charArray[Math.floor(Math.random() * charArray.length)];
    ctx.fillText(char, i * fontSize, drop * fontSize);

    if(drop * fontSize > matrixCanvas.height && Math.random() > 0.975) {
      drops[i] = 0;
    }
    drops[i]++;
  });
}

setInterval(drawMatrix, 35);

window.addEventListener('resize', function() {
  matrixCanvas.width = window.innerWidth;
  matrixCanvas.height = window.innerHeight;
});

async function generateSignature(password, timestamp, action, data = null) {
  try {
    const encoder = new TextEncoder();

    // Get SUPABASE_URL from currentSession or environment
    const supabaseUrl = currentSession.supabaseUrl ||
'https://legmizhfmslpdvijtw.supabase.co';

```

```

// ===== FIX THIS PART =====
// MASK the URL to match backend (use '***.supabase.co' instead of actual)
const maskedUrl = supabaseUrl.replace(/\/\.(.*?)\.supabase\.co/, '\/***.supabase.co');

// Create the message WITH MASKED SUPABASE_URL to match backend
let message = `${password}:${timestamp}:${action}:${maskedUrl}`;
// ===== END FIX =====

// Check if data should be included - FIXED to handle null properly
if (data !== null && data !== undefined) {
  // Check if it's an object with keys
  if (typeof data === 'object' && Object.keys(data).length > 0) {
    // Only include for non-read actions
    if (!['GET_ALL_TABLES', 'GET_TABLE'].includes(action)) {
      const dataString = JSON.stringify(data);
      message += `:${dataString}`;
    }
  } else if (typeof data === 'string' && data.trim() !== "") {
    if (!['GET_ALL_TABLES', 'GET_TABLE'].includes(action)) {
      message += `:${data}`;
    }
  }
}

console.log('🔒 FRONTEND FINAL SIGNATURE MESSAGE:', message.substring(0, 100) +
'...');
console.log('🔒 FRONTEND MASKED URL:', maskedUrl);

const messageData = encoder.encode(message);

// Use password as the HMAC key
const key = await crypto.subtle.importKey(
  'raw',
  encoder.encode(password),
  { name: 'HMAC', hash: 'SHA-256' },
  false,
  ['sign']
);

const signature = await crypto.subtle.sign('HMAC', key, messageData);
const hashArray = Array.from(new Uint8Array(signature));
const signatureHex = hashArray.map(b => b.toString(16).padStart(2, '0')).join("");

```

```

    return signatureHex;

} catch (error) {
    console.error('Signature generation failed:', error);
    return null;
}
}

// REPLACE THE generateIPValidator() FUNCTION WITH THIS:
async function generateIPValidator(timestamp) {
    try {
        // Get password from session or input
        const password = currentSession.token || document.getElementById('password').value;
        if (!password) {
            console.error('No password available for IP validator');
            return 'no-password-error';
        }

        // Use the provided timestamp (must match request body timestamp)
        if (!timestamp) {
            timestamp = new Date().toISOString();
        }

        // EXACT MATCH: password + first 10 chars of timestamp (YYYY-MM-DD)
        const message = password + timestamp.slice(0, 10);

        // Create SHA-256 hash
        const encoder = new TextEncoder();
        const data = encoder.encode(message);
        const hashBuffer = await crypto.subtle.digest('SHA-256', data);
        const hashArray = Array.from(new Uint8Array(hashBuffer));

        // Convert to hex string
        const hashHex = hashArray.map(b => b.toString(16).padStart(2, '0')).join("");

        // Take first 32 characters
        const result = hashHex.slice(0, 32);

        console.log('IP Validator Generated:', {
            passwordPreview: password.substring(0, 3) + '...',
            timestampSlice: timestamp.slice(0, 10),
            messagePreview: message.substring(0, 10) + '...',
            result: result
        });
    }
}

```

```

    return result;

} catch (error) {
    console.error('IP validator generation failed:', error);
    return 'fallback-' + Date.now().toString().slice(-8);
}
}

// =====
// API COMMUNICATION
// =====
async function makeSecureRequest(action, requestParams = {}) {
    const password = currentSession.token || document.getElementById('password').value;
    if (!password) {
        addLog('ERROR: No authentication token available', 'danger');
        return null;
    }

    const timestamp = new Date().toISOString();

    try {
        console.log('🔒 FRONTEND REQUEST:', { action, params: requestParams });

        // Generate signature - Handle different actions differently
        let signatureData = null;

        if (['GET_ALL_TABLES', 'GET_TABLE'].includes(action)) {
            // For these actions, don't include any data in signature
            signatureData = null;
        } else if (action === 'RAW_SQL') {
            // For SQL, include the query string in signature - BUT backend expects it in req.body.data
            // So we need to structure it as { query: "SQL here" } for signature
            signatureData = { query: requestParams.query };
        } else if (action === 'INSERT' || action === 'UPDATE' || action === 'DELETE') {
            // For INSERT/UPDATE/DELETE, include the data object
            signatureData = requestParams.data || requestParams;
        }

        const signature = await generateSignature(password, timestamp, action, signatureData);

        const ipValidator = await generateIPValidator(timestamp);

        if (!signature) {
            throw new Error('Signature generation failed');
        }
    }
}

```

```

}

// Build request - MATCH BACKEND EXPECTATIONS
const requestData = {
  password: password,
  timestamp: timestamp,
  signature: signature,
  action: action
};

// Add parameters based on action type (MATCHES BACKEND)
if (action === 'GET_TABLE') {
  requestData.table = requestParams.table;
  requestData.limit = requestParams.limit || 100;
}
else if (action === 'RAW_SQL') {
  // FIX: Backend expects query in req.body.query AND data in req.body.data for signature
  requestData.query = requestParams.query;
  requestData.data = { query: requestParams.query }; // Add to data for signature validation
}
else if (action === 'INSERT') {
  requestData.table = requestParams.table;
  requestData.data = requestParams.data;
}
else if (action === 'UPDATE') {
  requestData.table = requestParams.table;
  requestData.id = requestParams.id;
  requestData.data = requestParams.data;
}
else if (action === 'DELETE') {
  requestData.table = requestParams.table;
  requestData.id = requestParams.id;
}

console.log('🔒 FRONTEND SENDING:', requestData);

addLog(`API: Sending ${action} request...`, 'info');

const response = await fetch('/api/admin/database', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'X-IP-Validator': ipValidator
  },

```

```

        body: JSON.stringify(requestData)
    });

    const result = await response.json();

    if (response.ok && result.success) {
        addLog(`API: ${action} completed successfully`, 'success');
        return result;
    } else {
        console.error('API Error Response:', result);
        throw new Error(result.error || result.firewall || `API request failed: ${action}`);
    }
} catch (error) {
    addLog(`API ERROR: ${error.message}`, 'danger');
    return null;
}
}

// =====
// LOGIN / AUTHENTICATION
// =====
async function secureLogin() {
    const password = document.getElementById('password').value;
    if (!password) {
        addLog('ERROR: No password provided', 'danger');
        document.getElementById('authStatus').textContent = 'Authentication failed: No
password';
        document.getElementById('authStatus').style.color = '#ff3333';
        return;
    }

    addLog('SECURITY: Attempting authentication...', 'warning');
    document.getElementById('authStatus').textContent = 'Generating quantum signature...';
    document.getElementById('authStatus').style.color = '#ffaa00';

    const result = await makeSecureRequest('GET_ALL_TABLES');

    if (result) {
        // Authentication successful
        currentSession.authenticated = true;
        currentSession.token = password;
        currentSession.startTime = new Date();
        currentSession.supabaseUrl = result.supabaseUrl || 'Hidden for security';

        addLog('SUCCESS: Secure authentication completed', 'success');
    }
}

```

```

        document.getElementById('authStatus').textContent = 'Authentication successful!
Welcome, Administrator.';
        document.getElementById('authStatus').style.color = '#00cc66';

        // Update UI
        document.getElementById('loginPanel').style.display = 'none';
        document.getElementById('adminInterface').style.display = 'block';
        document.getElementById('sessionStatus').textContent = 'SESSION:
AUTHENTICATED';
        document.getElementById('sessionStatusDot').className = 'status-dot success';
        document.getElementById('supabaseUrl').textContent = currentSession.supabaseUrl;
        document.getElementById('sessionId').textContent = generateSessionId();

        // Load data
        if (result.tables && result.data) {
            currentSession.tables = result.tables;
            currentSession.data = result.data;
            populateTables(result.tables, result.data);
            document.getElementById('tableCount').textContent = result.tables.length;
            addLog(`Loaded ${result.tables.length} database tables`, 'success');
        }

        // Update operation terminal
        updateOperationTerminal('Ready for operations. Select a table to begin.');
```

} else {

```

        addLog('SECURITY: Authentication failed', 'danger');
        document.getElementById('authStatus').textContent = 'Authentication failed: Invalid
credentials';
        document.getElementById('authStatus').style.color = '#ff3333';

        // Security delay
        await new Promise(resolve => setTimeout(resolve, 2000));
        document.getElementById('password').value = '';
        document.getElementById('strengthMeter').style.width = '0%';
    }
}

function generateSessionId() {
    return Array.from(crypto.getRandomValues(new Uint8Array(8)))
        .map(b => b.toString(16).padStart(2, '0'))
        .join('');
}

```

```

function updateSessionTimer() {
  if (currentSession.startTime) {
    const now = new Date();
    const diff = Math.floor((now - currentSession.startTime) / 1000);
    const hours = Math.floor(diff / 3600);
    const minutes = Math.floor((diff % 3600) / 60);
    const seconds = diff % 60;
    document.getElementById('sessionTimer').textContent =
      `${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2,
'0')}:${seconds.toString().padStart(2, '0')}`;
  }
}

// =====
// TABLE MANAGEMENT
// =====
async function getAllTables() {
  addLog('DATABASE: Refreshing table list...', 'info');
  updateOperationTerminal('Refreshing table list...');

  const result = await makeSecureRequest('GET_ALL_TABLES');
  if (result) {
    currentSession.tables = result.tables || [];
    currentSession.data = result.data || {};
    populateTables(result.tables, result.data);
    document.getElementById('tableCount').textContent = result.tables.length;
    addLog(`Refreshed ${result.tables.length} tables`, 'success');
    updateOperationTerminal(`Loaded ${result.tables.length} tables successfully`);
  }
}

function populateTables(tables, data) {
  const tableList = document.getElementById('tableList');
  tableList.innerHTML = "";

  if (!tables || tables.length === 0) {
    tableList.innerHTML = '<div style="text-align: center; padding: 20px; color:
var(--text-secondary);">No tables found</div>';
    return;
  }

  tables.forEach(tableName => {
    const tableItem = document.createElement('div');
    tableItem.className = 'table-item';

```

```

        tableItem.onclick = () => selectTable(tableName);

        const rowCount = data && data[tableName] ? data[tableName].length : 0;
        const isActive = currentSession.currentTable === tableName;

        if (isActive) {
            tableItem.classList.add('active');
        }

        tableItem.innerHTML = `
            <span>${tableName}</span>
            <span class="row-count">${rowCount} rows</span>
        `;

        tableList.appendChild(tableItem);
    });
}

    async function makeSecureRequest(action, requestParams = {}) {
const password = currentSession.token || document.getElementById('password').value;
if (!password) {
    addLog('ERROR: No authentication token available', 'danger');
    return null;
}

const timestamp = new Date().toISOString();

try {
    console.log('🔒 FRONTEND REQUEST:', { action, params: requestParams });

    // Generate signature - for GET_TABLE, use null as data
    const signatureData = ['GET_ALL_TABLES', 'GET_TABLE'].includes(action) ? null :
requestParams;
    const signature = await generateSignature(password, timestamp, action, signatureData);

    const ipValidator = await generateIPValidator(timestamp);

    if (!signature) {
        throw new Error('Signature generation failed');
    }

    // Build request - MATCH BACKEND EXPECTATIONS
    const requestData = {
        password: password,

```

```

        timestamp: timestamp,
        signature: signature,
        action: action
    };

    // CRITICAL FIX: For GET_TABLE, add table and limit directly to requestData
    if (action === 'GET_TABLE') {
        requestData.table = requestParams.table;
        requestData.limit = requestParams.limit || 100;
    }
    // For other actions that need data field
    else if (['INSERT', 'UPDATE', 'DELETE', 'RAW_SQL'].includes(action)) {
        requestData.data = requestParams;
    }

    console.log('🔒 FRONTEND SENDING:', requestData);

    addLog(`API: Sending ${action} request...`, 'info');

    const response = await fetch('/api/admin/database', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-IP-Validator': ipValidator
        },
        body: JSON.stringify(requestData)
    });

    const result = await response.json();

    if (response.ok && result.success) {
        addLog(`API: ${action} completed successfully`, 'success');
        return result;
    } else {
        console.error('API Error Response:', result);
        throw new Error(result.error || result.firewall || `API request failed: ${action}`);
    }
} catch (error) {
    addLog(`API ERROR: ${error.message}`, 'danger');
    return null;
}

function displayTableData(data) {
    const table = document.getElementById('dataTable');

```

```

table.innerHTML = "";

if (!data || data.length === 0) {
  table.innerHTML = `
    <thead>
      <tr>
        <th style="width: 50px;">#</th>
        <th>Select</th>
        <th>Message</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td></td>
        <td style="text-align: center; color: var(--text-secondary);">No data
available</td>
      </tr>
    </tbody>
  `;
  return;
}

// Create headers from first object
const headers = Object.keys(data[0]);
const headerRow = document.createElement('tr');

// Add index and select headers
headerRow.innerHTML = '<th style="width: 50px;">#</th><th style="width:
60px;">Select</th>';

// Add data headers
headers.forEach(header => {
  const th = document.createElement('th');
  th.textContent = header;
  th.title = header;
  headerRow.appendChild(th);
});

table.appendChild(headerRow);

// Add data rows
const tbody = document.createElement('tbody');
data.forEach((row, index) => {

```

```

const tr = document.createElement('tr');
tr.dataset.rowIndex = index;
tr.dataset.rowId = row.id || index;

// Add index cell
const indexCell = document.createElement('td');
indexCell.textContent = index + 1;
tr.appendChild(indexCell);

// Add select checkbox
const selectCell = document.createElement('td');
const checkbox = document.createElement('input');
checkbox.type = 'checkbox';
checkbox.onclick = (e) => {
  e.stopPropagation();
  toggleRowSelection(index, checkbox.checked);
};
selectCell.appendChild(checkbox);
tr.appendChild(selectCell);

// Add data cells
headers.forEach(header => {
  const td = document.createElement('td');
  let cellValue = row[header];

  // Format the cell value
  if (cellValue === null || cellValue === undefined) {
    cellValue = 'NULL';
  } else if (typeof cellValue === 'object') {
    cellValue = JSON.stringify(cellValue);
  }

  cellValue = String(cellValue);
  if (cellValue.length > 50) {
    td.textContent = cellValue.substring(0, 50) + '...';
    td.title = cellValue;
  } else {
    td.textContent = cellValue;
  }

  tr.appendChild(td);
});

// Add click handler for row selection

```

```

        tr.onclick = (e) => {
            if (e.target.type !== 'checkbox') {
                checkbox.checked = !checkbox.checked;
                toggleRowSelection(index, checkbox.checked);
            }
        };

        tbody.appendChild(tr);
    });

    table.appendChild(tbody);
}

function toggleRowSelection(index, selected) {
    if (selected) {
        currentSession.selectedRows.add(index);
    } else {
        currentSession.selectedRows.delete(index);
    }

    // Update UI based on selection
    const hasSelection = currentSession.selectedRows.size > 0;
    document.getElementById('btnUpdate').disabled = !hasSelection;
    document.getElementById('btnDelete').disabled = !hasSelection;
    document.getElementById('btnView').disabled = !hasSelection;

    if (hasSelection) {
        updateOperationTerminal(`${currentSession.selectedRows.size} row(s) selected`);
    }
}

// =====
// DATA OPERATIONS
// =====
function showInsertModal() {
    if (!currentSession.currentTable) return;

    const form = document.getElementById('insertForm');
    form.innerHTML = '<div style="color: var(--text-secondary); margin-bottom: 15px;">Enter
values for new row:</div>';

    // Create form based on table structure (simplified - in real app, you'd fetch schema)
    const sampleRow = currentSession.currentData[0] || {};
    Object.keys(sampleRow).forEach(key => {

```

```

        const div = document.createElement('div');
        div.className = 'form-group';
        div.innerHTML = `
            <label class="form-label">${key}</label>
            <input type="text" class="form-input" id="insert_${key}" placeholder="Enter value
for ${key}">
        `;
        form.appendChild(div);
    });

    showModal('insertModal');
}

```

```

async function submitInsert() {
    const formData = {};
    const inputs = document.querySelectorAll('#insertForm .form-input');

    inputs.forEach(input => {
        const key = input.id.replace('insert_', '');
        formData[key] = input.value || null;
    });

    addLog(`INSERT: Adding row to ${currentSession.currentTable}...`, 'warning');
    updateOperationTerminal(`Inserting row into ${currentSession.currentTable}...`);

    const result = await makeSecureRequest('INSERT', {
        table: currentSession.currentTable,
        data: formData
    });

    if (result) {
        addLog(`INSERT: Row added successfully`, 'success');
        updateOperationTerminal('Row inserted successfully');
        closeModal('insertModal');
        await selectTable(currentSession.currentTable); // Refresh table
    }
}

```

```

function showUpdateModal() {
    if (!currentSession.currentTable || currentSession.selectedRows.size === 0) return;

    const selectedIndex = Array.from(currentSession.selectedRows)[0];
    const selectedRow = currentSession.currentData[selectedIndex];

```

```

const form = document.getElementById('updateForm');
form.innerHTML = `<div style="color: var(--text-secondary); margin-bottom:
15px;">Updating row ID: ${selectedRow.id || selectedIndex}</div>`;

Object.keys(selectedRow).forEach(key => {
  const div = document.createElement('div');
  div.className = 'form-group';
  const value = selectedRow[key];
  div.innerHTML = `
    <label class="form-label">${key}</label>
    <input type="text" class="form-input" id="update_${key}" value="${value || ""}">
  `;
  form.appendChild(div);
});

showModal('updateModal');
}

async function submitUpdate() {
  const selectedIndex = Array.from(currentSession.selectedRows)[0];
  const selectedRow = currentSession.currentData[selectedIndex];
  const rowId = selectedRow.id || selectedIndex;

  const formData = {};
  const inputs = document.querySelectorAll('#updateForm .form-input');

  inputs.forEach(input => {
    const key = input.id.replace('update_', '');
    formData[key] = input.value || null;
  });

  addLog(`UPDATE: Modifying row ${rowId} in ${currentSession.currentTable}...`,
'warning');
  updateOperationTerminal(`Updating row ${rowId}...`);

  const result = await makeSecureRequest('UPDATE', {
    table: currentSession.currentTable,
    id: rowId,
    data: formData
  });

  if (result) {
    addLog(`UPDATE: Row ${rowId} updated successfully`, 'success');
    updateOperationTerminal('Row updated successfully');
  }
}

```

```

        closeModal('updateModal');
        await selectTable(currentSession.currentTable); // Refresh table
    }
}

function showDeleteConfirm() {
    if (!currentSession.currentTable || currentSession.selectedRows.size === 0) return;

    const selectedCount = currentSession.selectedRows.size;
    document.getElementById('confirmMessage').textContent =
        `Are you sure you want to delete ${selectedCount} row(s) from
        ${currentSession.currentTable}? This action cannot be undone.`;

    window.currentAction = {
        type: 'DELETE',
        table: currentSession.currentTable,
        rowIds: Array.from(currentSession.selectedRows).map(idx =>
            currentSession.currentData[idx].id || idx
        )
    };

    showModal('confirmModal');
}

async function confirmAction() {
    const action = window.currentAction;
    if (!action) return;

    closeModal('confirmModal');

    if (action.type === 'DELETE') {
        addLog(`DELETE: Removing ${action.rowIds.length} row(s) from ${action.table}...`,
'danger');
        updateOperationTerminal(`Deleting ${action.rowIds.length} row(s)...`);

        // Delete each row (in real app, you might batch these)
        let successCount = 0;
        for (const rowId of action.rowIds) {
            const result = await makeSecureRequest('DELETE', {
                table: action.table,
                id: rowId
            });
            if (result) successCount++;
        }
    }
}

```

```

        if (successCount > 0) {
            addLog(`DELETE: ${successCount} row(s) removed successfully`, 'success');
            updateOperationTerminal(`${successCount} row(s) deleted`);
            await selectTable(currentSession.currentTable); // Refresh table
        }
    }

    window.currentAction = null;
}

function viewRowData() {
    if (!currentSession.currentTable || currentSession.selectedRows.size === 0) return;

    const selectedIndex = Array.from(currentSession.selectedRows)[0];
    const selectedRow = currentSession.currentData[selectedIndex];

    document.getElementById('viewData').textContent =
        JSON.stringify(selectedRow, null, 2);

    showModal('viewModal');
}

// =====
// SQL OPERATIONS
// =====
async function executeSQL() {
    const sql = document.getElementById('sqlEditor').value.trim();
    if (!sql) {
        addLog('ERROR: No SQL query provided', 'danger');
        return;
    }

    addLog(`SQL: Executing query...`, 'warning');
    updateOperationTerminal('Executing SQL query...');

    const result = await makeSecureRequest('RAW_SQL', {
        query: sql
    });

    if (result) {
        addLog(`SQL: Query executed successfully`, 'success');
        updateOperationTerminal('SQL query completed');
    }
}

```

```

    // Display results if any
    if (result.data && result.data.length > 0) {
        currentSession.currentData = result.data;
        displayTableData(result.data);
    }
}

function explainSQL() {
    const sql = document.getElementById('sqlEditor').value.trim();
    if (!sql) {
        addLog('ERROR: No SQL query provided', 'danger');
        return;
    }

    addLog(`SQL: Explaining query...`, 'info');
    updateOperationTerminal('Analyzing query execution plan...');

    // In a real implementation, this would call an EXPLAIN API
    setTimeout(() => {
        updateOperationTerminal('EXPLAIN analysis complete. Check logs for details.');
        addLog('SQL EXPLAIN: Query uses index scan (Estimated cost: 0.00..1.01)', 'info');
    }, 1000);
}

// =====
// EXPORT FUNCTIONS
// =====
function exportData(format) {
    if (!currentSession.currentData || currentSession.currentData.length === 0) {
        addLog('ERROR: No data to export', 'danger');
        return;
    }

    let content, mimeType, filename;

    if (format === 'csv') {
        content = convertToCSV(currentSession.currentData);
        mimeType = 'text/csv';
        filename = `${currentSession.currentTable}_${Date.now()}.csv`;
    } else {
        content = JSON.stringify(currentSession.currentData, null, 2);
        mimeType = 'application/json';
        filename = `${currentSession.currentTable}_${Date.now()}.json`;
    }

```

```

    }

    const blob = new Blob([content], { type: mimeType });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);

    addLog(`EXPORT: Data exported as ${format.toUpperCase()}`, 'success');
}

function convertToCSV(data) {
    if (data.length === 0) return "";

    const headers = Object.keys(data[0]);
    const rows = data.map(row =>
        headers.map(header => {
            let cell = row[header];
            if (cell === null || cell === undefined) cell = "";
            if (typeof cell === 'object') cell = JSON.stringify(cell);
            cell = String(cell);
            // Escape quotes and wrap in quotes if contains comma or quote
            if (cell.includes(',') || cell.includes('"') || cell.includes('\n')) {
                cell = `"${cell.replace(/"/g, '"')}"`;
            }
        })
        return cell;
    ).join(',');

    return [headers.join(','), ...rows].join('\n');
}

// =====
// UTILITY FUNCTIONS
// =====
function addLog(message, type = 'info') {
    const logs = document.getElementById('firewallLogs');
    const logEntry = document.createElement('div');
    logEntry.className = `log-entry ${type}`;

```

```

const now = new Date();
const timestamp = now.toISOString().replace('T', ' ').substring(0, 19);

logEntry.innerHTML = `
  <span class="log-timestamp">[${timestamp}]</span>
  <span>${message}</span>
`;

logs.appendChild(logEntry);
logs.scrollTop = logs.scrollHeight;
}

function updateOperationTerminal(message) {
  const terminal = document.getElementById('operationTerminal');
  const line = document.createElement('div');
  line.className = 'terminal-line';

  const now = new Date();
  const time = now.toTimeString().substring(0, 8);

  line.innerHTML = `
    <span class="terminal-prompt">${time} db@ops:~$</span>
    <span class="terminal-output">${message}</span>
  `;

  terminal.appendChild(line);
  terminal.scrollTop = terminal.scrollHeight;
  document.getElementById('operationStatus').textContent = message;
}

function showModal(modalId) {
  document.getElementById(modalId).style.display = 'flex';
}

function closeModal(modalId) {
  document.getElementById(modalId).style.display = 'none';
}

function clearSelection() {
  currentSession.selectedRows.clear();
  currentSession.currentTable = null;
  currentSession.currentData = null;

  document.getElementById('currentTable').textContent = 'None Selected';
}

```

```

document.getElementById('dataTable').innerHTML = `
    <thead>
        <tr>
            <th style="width: 50px;">#</th>
            <th>Select</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td colspan="10" style="text-align: center; padding: 40px; color:
var(--text-secondary);">
                Select a table to view data
            </td>
        </tr>
    </tbody>
`;

document.getElementById('btnInsert').disabled = true;
document.getElementById('btnUpdate').disabled = true;
document.getElementById('btnDelete').disabled = true;
document.getElementById('btnView').disabled = true;

updateOperationTerminal('Selection cleared');
addLog('Selection cleared', 'info');
}

function clearEditor() {
    document.getElementById('sqlEditor').value = "";
    addLog('SQL editor cleared', 'info');
}

async function refreshData() {
    if (currentSession.currentTable) {
        await selectTable(currentSession.currentTable);
    }
}

function resetInterface() {
    if (confirm('⚠️ EMERGENCY LOCKDOWN\n\nThis will immediately terminate the
session and clear all data.\n\nAre you sure?')) {
        addLog('SECURITY: Emergency lockdown initiated', 'danger');

        // Reset session
        currentSession = {

```

```

        authenticated: false,
        token: null,
        tables: [],
        currentTable: null,
        currentData: null,
        selectedRows: new Set(),
        startTime: null,
        supabaseUrl: null
    };

    // Reset UI
    document.getElementById('loginPanel').style.display = 'block';
    document.getElementById('adminInterface').style.display = 'none';
    document.getElementById('password').value = '';
    document.getElementById('strengthMeter').style.width = '0%';
    document.getElementById('sessionStatus').textContent = 'SESSION: NOT
AUTHENTICATED';
    document.getElementById('sessionStatusDot').className = 'status-dot';
    document.getElementById('authStatus').textContent = 'Waiting for credentials...';
    document.getElementById('authStatus').style.color = '';

    clearSelection();
    clearEditor();

    addLog('SYSTEM: Emergency lockdown complete. All sessions terminated.',
'danger');
    }
}

// Password strength indicator
document.getElementById('password').addEventListener('input', function() {
    const password = this.value;
    let strength = 0;

    if (password.length > 0) strength += 20;
    if (password.length >= 8) strength += 20;
    if (/[A-Z]/.test(password)) strength += 20;
    if (/0-9]/.test(password)) strength += 20;
    if (/^[A-Za-z0-9]/.test(password)) strength += 20;

    const meter = document.getElementById('strengthMeter');
    meter.style.width = strength + '%';
    meter.style.background = strength < 40 ? '#ff3333' :
        strength < 80 ? '#ffaa00' : '#00cc66';

```

```
});  
</script>  
</body>  
</html>
```

public/home/editor.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Pack Editor - WebAssembly Support</title>
```

```
  <link rel="stylesheet" data-name="vs/editor/editor.main"
```

```
href="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/editor/editor.main.min.c  
ss">
```

```
  <style>
```

```
    * {  
      margin: 0;  
      padding: 0;  
      box-sizing: border-box;  
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
    }
```

```
    body {  
      background-color: #1e1e1e;  
      color: #d4d4d4;  
      display: flex;  
      flex-direction: column;  
      height: 100vh;  
      overflow: hidden;  
    }
```

```
    header {  
      background-color: #252526;  
      padding: 15px 20px;  
      display: flex;  
      justify-content: space-between;  
      align-items: center;  
      border-bottom: 1px solid #3e3e42;  
    }
```

```
    .logo {  
      font-size: 1.5rem;  
      font-weight: bold;
```

```
    color: #569cd6;
}

.controls {
  display: flex;
  gap: 10px;
}

button {
  background-color: #0e639c;
  color: white;
  border: none;
  padding: 8px 16px;
  border-radius: 4px;
  cursor: pointer;
  font-weight: 500;
  transition: background-color 0.2s;
}

button:hover {
  background-color: #1177bb;
}

button.secondary {
  background-color: #3e3e42;
}

button.secondary:hover {
  background-color: #4a4a4f;
}

button.success {
  background-color: #388a34;
}

button.success:hover {
  background-color: #3fa33a;
}

button:disabled {
  background-color: #3e3e42;
  cursor: not-allowed;
  opacity: 0.6;
}
```

```
.container {
  display: flex;
  flex: 1;
  overflow: hidden;
}

.sidebar {
  width: 280px;
  background-color: #252526;
  border-right: 1px solid #3e3e42;
  padding: 20px;
  overflow-y: auto;
}

.sidebar h3 {
  margin-bottom: 15px;
  color: #cccccc;
  font-size: 1rem;
  font-weight: 500;
}

.file-list {
  list-style-type: none;
  margin-bottom: 30px;
  max-height: 200px;
  overflow-y: auto;
}

.file-list li {
  padding: 8px 12px;
  border-radius: 4px;
  margin-bottom: 5px;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.file-list li:hover {
  background-color: #2a2d2e;
}

.file-list li.active {
```

```
    background-color: #094771;
}
```

```
.file-icon {
    color: #c586c0;
    font-size: 0.9rem;
}
```

```
.json-file .file-icon {
    color: #f5d76e;
}
```

```
.js-file .file-icon {
    color: #4ec9b0;
}
```

```
.wasm-file .file-icon {
    color: #d16969;
}
```

```
.file-actions {
    display: flex;
    gap: 5px;
    opacity: 0;
    transition: opacity 0.2s;
}
```

```
.file-list li:hover .file-actions {
    opacity: 1;
}
```

```
.file-action-btn {
    background: none;
    border: none;
    color: #858585;
    padding: 2px 5px;
    font-size: 0.8rem;
}
```

```
.file-action-btn:hover {
    color: #d4d4d4;
}
```

```
.wasm-section {
```

```
    margin-top: 20px;
    border-top: 1px solid #3e3e42;
    padding-top: 20px;
}
```

```
.wasm-info {
    background-color: #2a2d2e;
    padding: 10px;
    border-radius: 4px;
    margin-bottom: 15px;
    font-size: 0.85rem;
}
```

```
.wasm-info.warning {
    background-color: #5c2d0c;
    border-left: 3px solid #f5d76e;
}
```

```
.wasm-info.success {
    background-color: #2d5c0c;
    border-left: 3px solid #3fa33a;
}
```

```
.upload-area {
    border: 2px dashed #3e3e42;
    border-radius: 8px;
    padding: 20px;
    text-align: center;
    cursor: pointer;
    transition: border-color 0.2s;
    margin-bottom: 15px;
}
```

```
.upload-area:hover {
    border-color: #0e639c;
}
```

```
.upload-area.dragover {
    border-color: #569cd6;
    background-color: rgba(86, 156, 214, 0.1);
}
```

```
.upload-icon {
    font-size: 2rem;
}
```

```
    color: #569cd6;
    margin-bottom: 10px;
}
```

```
.upload-area p {
    margin-bottom: 5px;
}
```

```
.upload-hint {
    font-size: 0.8rem;
    color: #858585;
}
```

```
.wasm-list {
    list-style-type: none;
    margin-top: 10px;
    max-height: 150px;
    overflow-y: auto;
}
```

```
.wasm-list li {
    padding: 8px 12px;
    background-color: #2a2d2e;
    border-radius: 4px;
    margin-bottom: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    font-size: 0.9rem;
}
```

```
.wasm-size {
    color: #858585;
    font-size: 0.8rem;
}
```

```
.editor-container {
    flex: 1;
    display: flex;
    flex-direction: column;
    overflow: hidden;
}
```

```
.editor-tabs {
```

```
    display: flex;
    background-color: #2d2d30;
    border-bottom: 1px solid #3e3e42;
    overflow-x: auto;
}
```

```
.tab {
    padding: 10px 20px;
    border-right: 1px solid #3e3e42;
    cursor: pointer;
    white-space: nowrap;
    display: flex;
    align-items: center;
    gap: 8px;
}
```

```
.tab:hover {
    background-color: #2a2d2e;
}
```

```
.tab.active {
    background-color: #1e1e1e;
}
```

```
.close-tab {
    margin-left: 5px;
    opacity: 0.7;
}
```

```
.close-tab:hover {
    opacity: 1;
}
```

```
#editor {
    flex: 1;
    overflow: hidden;
}
```

```
.config-panel {
    margin-top: 20px;
}
```

```
.config-item {
    margin-bottom: 15px;
}
```

```
}
```

```
label {  
  display: block;  
  margin-bottom: 5px;  
  font-size: 0.9rem;  
  color: #cccccc;  
}
```

```
input[type="text"], select {  
  width: 100%;  
  padding: 8px 12px;  
  background-color: #3e3e42;  
  border: 1px solid #565656;  
  border-radius: 4px;  
  color: #d4d4d4;  
}
```

```
input[type="text"]:focus, select:focus {  
  outline: none;  
  border-color: #007acc;  
}
```

```
.checkbox-group {  
  display: flex;  
  align-items: center;  
  gap: 8px;  
  margin-top: 5px;  
}
```

```
input[type="checkbox"] {  
  accent-color: #0e639c;  
}
```

```
.publish-modal {  
  position: fixed;  
  top: 0;  
  left: 0;  
  width: 100%;  
  height: 100%;  
  background-color: rgba(0, 0, 0, 0.8);  
  display: flex;  
  justify-content: center;  
  align-items: center;
```

```
    z-index: 1000;
    display: none;
}

.modal-content {
    background-color: #252526;
    border-radius: 8px;
    padding: 25px;
    width: 90%;
    max-width: 500px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.5);
}

.modal-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.modal-header h2 {
    color: #569cd6;
    font-size: 1.5rem;
}

.close-modal {
    font-size: 1.8rem;
    cursor: pointer;
    color: #cccccc;
}

.close-modal:hover {
    color: white;
}

.publish-result {
    background-color: #1e1e1e;
    padding: 15px;
    border-radius: 4px;
    margin-top: 20px;
    font-family: monospace;
    font-size: 0.9rem;
    white-space: pre-wrap;
    word-break: break-all;
}
```

```
    display: none;
}

.result-item {
    margin-bottom: 10px;
}

.result-label {
    color: #9cdcfe;
    font-weight: bold;
}

.result-value {
    color: #ce9178;
}

.result-value a {
    color: #569cd6;
    text-decoration: none;
}

.result-value a:hover {
    text-decoration: underline;
}

.loading {
    display: none;
    text-align: center;
    margin: 20px 0;
    color: #569cd6;
}

.spinner {
    border: 3px solid rgba(86, 156, 214, 0.3);
    border-radius: 50%;
    border-top: 3px solid #569cd6;
    width: 30px;
    height: 30px;
    animation: spin 1s linear infinite;
    margin: 0 auto 10px;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
```

```

    100% { transform: rotate(360deg); }
  }

.notification {
  position: fixed;
  bottom: 20px;
  right: 20px;
  padding: 12px 20px;
  background-color: #0c7b93;
  color: white;
  border-radius: 4px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
  z-index: 1001;
  transform: translateY(100px);
  opacity: 0;
  transition: transform 0.3s, opacity 0.3s;
}

.notification.show {
  transform: translateY(0);
  opacity: 1;
}

.notification.error {
  background-color: #a12c2c;
}

.notification.success {
  background-color: #388a34;
}

footer {
  padding: 10px 20px;
  background-color: #252526;
  border-top: 1px solid #3e3e42;
  font-size: 0.8rem;
  color: #858585;
  text-align: center;
}
</style>
</head>
<body>
  <header>
    <div class="logo">Pack Editor</div>

```

```
<div class="controls">
  <button id="newFileBtn" class="secondary">New File</button>
  <button id="saveBtn">Save</button>
  <button id="publishBtn" class="success">Publish Pack</button>
</div>
</header>

<div class="container">
  <div class="sidebar">
    <h3>Files</h3>
    <ul class="file-list" id="fileList">
      <!-- Files will be added here dynamically -->
    </ul>

    <div class="wasm-section">
      <h3>WebAssembly Files</h3>

      <div class="wasm-info warning" id="wasmInfo">
        <strong>Important:</strong> WASM files cannot be edited directly. You can only
upload valid WebAssembly binary files (.wasm). These will be served from the CDN.
      </div>

      <div class="upload-area" id="uploadArea">
        <div class="upload-icon">⬆</div>
        <p>Drag & drop WASM files here</p>
        <p class="upload-hint">or click to browse</p>
        <p class="upload-hint">Only .wasm files accepted</p>
      </div>

      <ul class="wasm-list" id="wasmList">
        <!-- WASM files will be listed here -->
      </ul>
    </div>

    <div class="config-panel">
      <h3>Pack Configuration</h3>

      <div class="config-item">
        <label for="packName">Pack Name</label>
        <input type="text" id="packName" placeholder="my-awesome-pack"
value="my-awesome-pack">
      </div>

      <div class="config-item">
```

```
<label for="packType">Package Type</label>
<select id="packType">
  <option value="module">Module</option>
  <option value="library">Library</option>
  <option value="template">Template</option>
  <option value="plugin">Plugin</option>
</select>
</div>
```

```
<div class="config-item">
  <label>Visibility</label>
  <div class="checkbox-group">
    <input type="checkbox" id="isPublic" checked>
    <label for="isPublic">Public Package</label>
  </div>
</div>
</div>
</div>
```

```
<div class="editor-container">
  <div class="editor-tabs" id="editorTabs">
    <!-- Editor tabs will be added here dynamically -->
  </div>
  <div id="editor"></div>
</div>
</div>
```

```
<div class="publish-modal" id="publishModal">
  <div class="modal-content">
    <div class="modal-header">
      <h2>Publish Pack</h2>
      <div class="close-modal" id="closeModal">&times;</div>
    </div>
```

<p>Ready to publish your pack? This will upload all files and generate a unique URL for your package.</p>

```
<div id="validationResults">
  <!-- Validation results will appear here -->
</div>
```

```
<div class="loading" id="publishLoading">
  <div class="spinner"></div>
  <p>Publishing your pack...</p>
```

```

    </div>

    <div class="publish-result" id="publishResult">
      <!-- Publish results will be displayed here -->
    </div>

    <div style="display: flex; gap: 10px; margin-top: 20px;">
      <button id="confirmPublishBtn" class="success" style="flex: 1;">Publish
Now</button>
      <button id="cancelPublishBtn" class="secondary">Cancel</button>
    </div>
  </div>
</div>

<div class="notification" id="notification">Message</div>

<footer>
  Pack Editor - Edit and publish packages with WebAssembly support
</footer>

<!-- Monaco Editor loader -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/loader.min.js"></script>
<script>
  // Monaco editor configuration
  require.config({ paths: { vs:
'https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs' } });

  // Initialize app state
  const appState = {
    files: {
      'package.json': {
        name: 'package.json',
        content: JSON.stringify({
          name: 'my-awesome-pack',
          version: '1.0.0',
          type: 'module',
          description: 'My awesome package with WASM support',
          main: 'index.js',
          author: '',
          license: 'MIT'
        }, null, 2),
        language: 'json',
        editable: true
      }
    }
  };

```

```
    },
    'index.js': {
      name: 'index.js',
      content: `// Main entry point for your package

export function helloWorld() {
  console.log('Hello from my awesome pack!');
  return 'Hello World!';
}

// Add your code here
export const version = '1.0.0';

// Example utility function
export function sum(a, b) {
  return a + b;
}`,
      language: 'javascript',
      editable: true
    },
    'README.md': {
      name: 'README.md',
      content: `# My Awesome Pack`
    }
  }
}
```

This is a sample package created with Pack Editor.

Installation

```
```bash
pack install my-awesome-pack
```
```

Usage

```
```javascript
import { helloWorld } from 'my-awesome-pack';

helloWorld(); // Outputs: Hello from my awesome pack!
```
```

API

```
#### helloWorld()
Prints a greeting message and returns "Hello World!"
```

sum(a, b)

Returns the sum of two numbers

License

MIT

`

```
    language: 'markdown',
    editable: true
  }
},
wasmFiles: {},
activeFile: 'package.json',
editor: null,
packName: 'my-awesome-pack',
packType: 'module',
isPublic: true
};

// DOM Elements
const fileList = document.getElementById('fileList');
const editorTabs = document.getElementById('editorTabs');
const editorContainer = document.getElementById('editor');
const packNameInput = document.getElementById('packName');
const packTypeSelect = document.getElementById('packType');
const isPublicCheckbox = document.getElementById('isPublic');
const publishModal = document.getElementById('publishModal');
const closeModal = document.getElementById('closeModal');
const cancelPublishBtn = document.getElementById('cancelPublishBtn');
const confirmPublishBtn = document.getElementById('confirmPublishBtn');
const publishResult = document.getElementById('publishResult');
const publishLoading = document.getElementById('publishLoading');
const notification = document.getElementById('notification');
const uploadArea = document.getElementById('uploadArea');
const wasmList = document.getElementById('wasmList');
const wasmInfo = document.getElementById('wasmInfo');
const validationResults = document.getElementById('validationResults');

// File input for WASM uploads (hidden)
const fileInput = document.createElement('input');
fileInput.type = 'file';
fileInput.accept = '.wasm';
fileInput.multiple = true;
```

```

fileInput.style.display = 'none';
document.body.appendChild(fileInput);

// Initialize Monaco Editor
require(['vs/editor/editor.main'], function() {
  // Create editor instance
  appState.editor = monaco.editor.create(editorContainer, {
    value: appState.files['package.json'].content,
    language: 'json',
    theme: 'vs-dark',
    minimap: { enabled: true },
    scrollBeyondLastLine: false,
    automaticLayout: true,
    fontSize: 14,
    lineNumbers: 'on',
    roundedSelection: false,
    scrollbar: {
      vertical: 'auto',
      horizontal: 'auto'
    },
    readOnly: false
  });

  // Update app state when editor content changes
  appState.editor.onDidChangeModelContent(() => {
    const activeFile = appState.activeFile;
    if (activeFile && appState.files[activeFile]) {
      appState.files[activeFile].content = appState.editor.getValue();

      // If this is package.json, update the pack name if changed
      if (activeFile === 'package.json') {
        try {
          const packageJson = JSON.parse(appState.editor.getValue());
          if (packageJson.name && packageJson.name !== appState.packName) {
            appState.packName = packageJson.name;
            packNameInput.value = packageJson.name;
          }
          if (packageJson.type && packageJson.type !== appState.packType) {
            appState.packType = packageJson.type;
            packTypeSelect.value = packageJson.type;
          }
        } catch (e) {
          // Invalid JSON, ignore
        }
      }
    }
  });
});

```

```

    }
  }
});

// Initialize UI
renderFileList();
renderEditorTabs();
setupEventListeners();
});

// Render the file list in the sidebar
function renderFileList() {
  fileList.innerHTML = "";

  Object.keys(appState.files).forEach(fileName => {
    const file = appState.files[fileName];
    const li = document.createElement('li');
    li.className = `file-item ${fileName === appState.activeFile ? 'active' : ''}
    ${file.language}-file`;

    const fileExt = fileName.split('.').pop();
    const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

    li.innerHTML = `
      <div>
        <span class="file-icon">${icon}</span>
        <span>${fileName}</span>
      </div>
      <div class="file-actions">
        <button class="file-action-btn" data-file="${fileName}"
title="Rename">✏️</button>
        ${fileName !== 'package.json' && fileName !== 'index.js' ?
        `<button class="file-action-btn" data-file="${fileName}"
title="Delete">🗑️</button>` : ""}
      </div>
    `;

    li.addEventListener('click', (e) => {
      if (!e.target.closest('.file-action-btn')) {
        switchFile(fileName);
      }
    });

    fileList.appendChild(li);
  });
}

```

```

});

// Add event listeners for file actions
document.querySelectorAll('.file-action-btn').forEach(btn => {
  btn.addEventListener('click', (e) => {
    e.stopPropagation();
    const fileName = btn.dataset.file;

    if (btn.textContent.includes('✎')) {
      renameFile(fileName);
    } else if (btn.textContent.includes('🗑')) {
      deleteFile(fileName);
    }
  });
});
});
}

// Render WASM file list
function renderWasmList() {
  wasmList.innerHTML = "";

  if (Object.keys(appState.wasmFiles).length === 0) {
    const li = document.createElement('li');
    li.textContent = 'No WASM files uploaded';
    li.style.color = '#858585';
    li.style.fontStyle = 'italic';
    wasmList.appendChild(li);
    wasmInfo.className = 'wasm-info warning';
    wasmInfo.innerHTML = '<strong>Important:</strong> WASM files cannot be edited directly. You can only upload valid WebAssembly binary files (.wasm). These will be served from the CDN.';
  } else {
    Object.keys(appState.wasmFiles).forEach(fileName => {
      const file = appState.wasmFiles[fileName];
      const li = document.createElement('li');

      li.innerHTML = `
        <div>
          <span class="file-icon">⚡</span>
          <span>${fileName}</span>
        </div>
        <div>
          <span class="wasm-size">${formatFileSize(file.size)}</span>

```

```

        <button class="file-action-btn" data-wasm="${fileName}"
title="Remove">🗑️ </button>
    </div>
    `;

    wasmList.appendChild(li);
});

wasmInfo.className = 'wasm-info success';
wasmInfo.innerHTML = `<strong>✓ Ready:</strong>
${Object.keys(appState.wasmFiles).length} WASM file(s) uploaded and validated.`;

// Add event listeners for remove buttons
document.querySelectorAll('[data-wasm]').forEach(btn => {
    btn.addEventListener('click', (e) => {
        const fileName = btn.dataset.wasm;
        deleteWasmFile(fileName);
    });
});
}
}

// Render editor tabs
function renderEditorTabs() {
    editorTabs.innerHTML = "";

    Object.keys(appState.files).forEach(fileName => {
        const tab = document.createElement('div');
        tab.className = `tab ${fileName === appState.activeFile ? 'active' : ''}`;
        tab.dataset.file = fileName;

        const fileExt = fileName.split('.').pop();
        const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

        tab.innerHTML = `
            <span class="file-icon">${icon}</span>
            <span>${fileName}</span>
            ${fileName !== 'package.json' && fileName !== 'index.js' ?
                '<span class="close-tab" data-file="' + fileName + '">✕</span>' : ''}
            `;

        tab.addEventListener('click', (e) => {
            if (!e.target.classList.contains('close-tab')) {
                switchFile(fileName);
            }
        });
    });
}

```

```

    }
  });

  editorTabs.appendChild(tab);
});

// Add event listeners for close tab buttons
document.querySelectorAll('.close-tab').forEach(btn => {
  btn.addEventListener('click', (e) => {
    e.stopPropagation();
    const fileName = btn.dataset.file;
    deleteFile(fileName);
  });
});
}

// Switch to a different file
function switchFile(fileName) {
  if (!appState.files[fileName]) return;

  appState.activeFile = fileName;
  const file = appState.files[fileName];

  // Update editor content and language
  appState.editor.setValue(file.content);
  monaco.editor.setModelLanguage(appState.editor.getModel(), file.language);

  // Update UI
  renderFileList();
  renderEditorTabs();
}

// Create a new file
function createNewFile() {
  const fileName = prompt('Enter file name (with extension, e.g., utils.js):', 'newfile.js');
  if (!fileName) return;

  if (appState.files[fileName] || appState.wasmFiles[fileName]) {
    showNotification('File already exists!', 'error');
    return;
  }

  const fileExt = fileName.split('.').pop();
  let language = 'plaintext';

```

```

let content = "";

switch (fileExt) {
  case 'js':
    language = 'javascript';
    content = `// ${fileName}\n\n`;
    break;
  case 'json':
    language = 'json';
    content = '{\n  \n}\n';
    break;
  case 'md':
    language = 'markdown';
    content = `# ${fileName}\n\n`;
    break;
  case 'html':
    language = 'html';
    content = `<!DOCTYPE html>\n<html>\n<head>\n
<title></title>\n</head>\n<body>\n  \n</body>\n</html>\n`;
    break;
  case 'css':
    language = 'css';
    content = `/* ${fileName} */\n\n`;
    break;
}

appState.files[fileName] = {
  name: fileName,
  content,
  language,
  editable: true
};

switchFile(fileName);
showNotification(`Created ${fileName}`, 'success');
}

// Rename a file
function renameFile(oldName) {
  const newName = prompt('Enter new file name:', oldName);
  if (!newName || newName === oldName) return;

  if (appState.files[newName] || appState.wasmFiles[newName]) {
    showNotification('File already exists!', 'error');
  }
}

```

```

        return;
    }

    appState.files[newName] = {
        ...appState.files[oldName],
        name: newName
    };

    delete appState.files[oldName];

    if (appState.activeFile === oldName) {
        appState.activeFile = newName;
    }

    renderFileList();
    renderEditorTabs();
    showNotification(`Renamed ${oldName} to ${newName}`, 'success');
}

// Delete a file
function deleteFile(fileName) {
    if (fileName === 'package.json' || fileName === 'index.js') {
        showNotification('Cannot delete required files!', 'error');
        return;
    }

    if (confirm(`Are you sure you want to delete ${fileName}?`)) {
        delete appState.files[fileName];

        if (appState.activeFile === fileName) {
            const remainingFiles = Object.keys(appState.files);
            if (remainingFiles.length > 0) {
                switchFile(remainingFiles[0]);
            }
        }

        renderFileList();
        renderEditorTabs();
        showNotification(`Deleted ${fileName}`, 'success');
    }
}

// Handle WASM file upload
async function handleWasmUpload(files) {

```

```

for (const file of files) {
  if (!file.name.endsWith('.wasm')) {
    showNotification(`Skipped ${file.name}: Not a .wasm file`, 'error');
    continue;
  }

  // Validate WASM file
  const isValid = await validateWasmFile(file);
  if (!isValid) {
    showNotification(`Skipped ${file.name}: Invalid WebAssembly file`, 'error');
    continue;
  }

  // Read file as ArrayBuffer
  const reader = new FileReader();
  reader.onload = async (e) => {
    const arrayBuffer = e.target.result;

    // Store WASM file
    appState.wasmFiles[file.name] = {
      name: file.name,
      size: file.size,
      data: arrayBuffer
    };

    renderWasmList();
    showNotification(`Uploaded ${file.name} (${formatFileSize(file.size)})`, 'success');
  };

  reader.readAsArrayBuffer(file);
}

// Validate a WASM file
async function validateWasmFile(file) {
  try {
    // Basic WASM validation - check magic number
    const arrayBuffer = await file.slice(0, 8).arrayBuffer();
    const header = new Uint8Array(arrayBuffer);

    // WASM magic number: \0asm
    const wasmMagic = [0x00, 0x61, 0x73, 0x6D];

    for (let i = 0; i < 4; i++) {

```

```

        if (header[i] !== wasmMagic[i]) {
            return false;
        }
    }

    return true;
} catch (error) {
    console.error('WASM validation error:', error);
    return false;
}
}

// Delete a WASM file
function deleteWasmFile(fileName) {
    if (confirm(`Are you sure you want to remove ${fileName}?`)) {
        delete appState.wasmFiles[fileName];
        renderWasmList();
        showNotification(`Removed ${fileName}`, 'success');
    }
}

// Format file size
function formatFileSize(bytes) {
    if (bytes === 0) return '0 Bytes';
    const k = 1024;
    const sizes = ['Bytes', 'KB', 'MB', 'GB'];
    const i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i];
}

// Show notification
function showNotification(message, type = 'info') {
    notification.textContent = message;
    notification.className = `notification ${type}`;
    notification.classList.add('show');

    setTimeout(() => {
        notification.classList.remove('show');
    }, 3000);
}

// Validate package before publishing
function validatePackage() {
    const errors = [];

```

```

const warnings = [];

// Check package.json
try {
  const packageJson = JSON.parse(appState.files['package.json'].content);

  if (!packageJson.name || packageJson.name.trim() === '') {
    errors.push('Package name is required in package.json');
  }

  if (!packageJson.version) {
    warnings.push('Consider adding a version field to package.json');
  }

  if (!packageJson.main) {
    warnings.push('Consider specifying a main entry point in package.json');
  }
} catch (e) {
  errors.push('Invalid JSON in package.json');
}

// Check if index.js exists
if (!appState.files['index.js']) {
  warnings.push('index.js is recommended as the main entry point');
}

// Check for invalid file names
Object.keys(appState.files).forEach(fileName => {
  if (fileName.includes(' ')) {
    warnings.push(`File "${fileName}" contains spaces, consider using hyphens or
underscores`);
  }
});

return { errors, warnings };
}

// Publish package to API - FIXED FOR API EXPECTATIONS
async function publishPackage() {
  // Validate package first
  const validation = validatePackage();

  if (validation.errors.length > 0) {
    showNotification('Cannot publish: ' + validation.errors[0], 'error');
  }
}

```

```

    return;
}

// Show loading state
publishLoading.style.display = 'block';
validationResults.innerHTML = "";

try {
    // Prepare package.json
    const packageJson = JSON.parse(appState.files['package.json'].content);
    packageJson.name = appState.packName;
    packageJson.type = appState.packType;

    // Prepare all files for upload - SIMPLE OBJECT AS API EXPECTS
    const files = {};

    // Add regular files - just the content as string
    Object.keys(appState.files).forEach(fileName => {
        files[fileName] = appState.files[fileName].content;
    });

    // Add WASM files - convert ArrayBuffer to base64 string
    Object.keys(appState.wasmFiles).forEach(fileName => {
        const wasmFile = appState.wasmFiles[fileName];
        // Convert ArrayBuffer to base64 string
        const bytes = new Uint8Array(wasmFile.data);
        let binary = "";
        for (let i = 0; i < bytes.byteLength; i++) {
            binary += String.fromCharCode(bytes[i]);
        }
        files[fileName] = btoa(binary); // Just the base64 string, not an object
    });

    // Prepare request body - EXACTLY AS API EXPECTS
    const requestBody = {
        name: appState.packName,
        packJson: JSON.stringify(packageJson), // Already a string in your API
        files: files, // Simple object: {filename: contentString}
        isPublic: appState.isPublic
    };

    console.log('Sending to API:', {
        name: requestBody.name,
        packJson: requestBody.packJson.substring(0, 100) + '...',

```

```
        fileCount: Object.keys(requestBody.files).length,
        fileNames: Object.keys(requestBody.files)
    });

    // Call publish API
    const response = await fetch('/api/publish.js', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(requestBody)
    });

    const responseText = await response.text();
    console.log('API Response:', responseText);

    if (!response.ok) {
        throw new Error(`HTTP ${response.status}: ${response.statusText}. Response:
    ${responseText}`);
    }

    const result = JSON.parse(responseText);

    // Hide loading, show results
    publishLoading.style.display = 'none';
    publishResult.style.display = 'block';

    // Display results
    publishResult.innerHTML = `
        <div class="result-item">
            <span class="result-label">✓ Published successfully!</span>
        </div>
        <div class="result-item">
            <span class="result-label">Pack ID:</span>
            <span class="result-value">${result.packId}</span>
        </div>
        <div class="result-item">
            <span class="result-label">CDN URL:</span>
            <span class="result-value"><a href="${result.cdnUrl}"
target="_blank">${result.cdnUrl}</a></span>
        </div>
        <div class="result-item">
            <span class="result-label">Worker URL:</span>
```

```

        <span class="result-value"><a href="${result.workerUrl}"
target="_blank">${result.workerUrl}</a></span>
    </div>
    <div class="result-item">
        <span class="result-label">Install Command:</span>
        <span class="result-value">${result.installCommand}</span>
    </div>
    ${result.encryptedKey ? `
    <div class="result-item">
        <span class="result-label">Encryption Key:</span>
        <span class="result-value">${result.encryptedKey}</span>
    </div>
    <div class="result-item">
        <span class="result-label">⚠ Save this key! It won't be shown again.</span>
    </div>
    ` : ""}
    <div class="result-item">
        <button id="exploreBtn" class="success" style="width: 100%; margin-top:
10px;">Explore Package</button>
    </div>
    `;

```

```

// Add event listener for explore button
document.getElementById('exploreBtn').addEventListener('click', () => {
    window.location.href = 'explore.html';
});

} catch (error) {
    publishLoading.style.display = 'none';
    showNotification(`Publish failed: ${error.message}`, 'error');
    console.error('Publish error:', error);
}
}

// Setup event listeners
function setupEventListeners() {
    // Pack configuration
    packNameInput.addEventListener('input', (e) => {
        appState.packName = e.target.value;
    });

    packTypeSelect.addEventListener('change', (e) => {
        appState.packType = e.target.value;
    });
}

```

```

isPublicCheckbox.addEventListener('change', (e) => {
  appState.isPublic = e.target.checked;
});

// New file button
document.getElementById('newFileBtn').addEventListener('click', createNewFile);

// Save button
document.getElementById('saveBtn').addEventListener('click', () => {
  showNotification('All changes saved locally', 'success');
});

// Publish button
document.getElementById('publishBtn').addEventListener('click', () => {
  // Validate before showing modal
  const validation = validatePackage();

  validationResults.innerHTML = "";

  if (validation.errors.length > 0) {
    validationResults.innerHTML = `
      <div style="background-color: #5c2d0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
        <strong>Errors:</strong>
        <ul style="margin: 5px 0 0 20px;">
          ${validation.errors.map(err => `<li>${err}</li>`).join("")}
        </ul>
      </div>
    `;
  } else if (validation.warnings.length > 0) {
    validationResults.innerHTML = `
      <div style="background-color: #5c5c0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
        <strong>Warnings:</strong>
        <ul style="margin: 5px 0 0 20px;">
          ${validation.warnings.map(warn => `<li>${warn}</li>`).join("")}
        </ul>
      </div>
    `;
  }

  publishModal.style.display = 'flex';
  publishResult.style.display = 'none';

```

```

        publishLoading.style.display = 'none';
    });

    // Modal controls
    closeModal.addEventListener('click', () => {
        publishModal.style.display = 'none';
    });

    cancelPublishBtn.addEventListener('click', () => {
        publishModal.style.display = 'none';
    });

    confirmPublishBtn.addEventListener('click', publishPackage);

    // WASM file upload
    uploadArea.addEventListener('click', () => {
        fileInput.click();
    });

    fileInput.addEventListener('change', (e) => {
        handleWasmUpload(Array.from(e.target.files));
        fileInput.value = "";
    });

    // Drag and drop for WASM files
    uploadArea.addEventListener('dragover', (e) => {
        e.preventDefault();
        uploadArea.classList.add('dragover');
    });

    uploadArea.addEventListener('dragleave', () => {
        uploadArea.classList.remove('dragover');
    });

    uploadArea.addEventListener('drop', (e) => {
        e.preventDefault();
        uploadArea.classList.remove('dragover');

        const files = Array.from(e.dataTransfer.files);
        handleWasmUpload(files);
    });
}

// Initialize WASM list

```

```

renderWasmList();

// Close modal when clicking outside
window.addEventListener('click', (e) => {
  if (e.target === publishModal) {
    publishModal.style.display = 'none';
  }
});
</script>
</body>
</html>

```

public/home/explore.html

```

<!DOCTYPE html>
<html>
<head>
  <title>Explore Packs - PackCDN</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body { margin: 0; font-family: -apple-system, BlinkMacSystemFont, sans-serif; background:
    #f8f9fa; }
    .container { max-width: 1200px; margin: 0 auto; padding: 20px; }
    header { background: white; padding: 20px; border-radius: 10px; margin-bottom: 30px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1); }
    .search-box { display: flex; gap: 10px; margin-top: 20px; }
    .search-box input { flex: 1; padding: 12px; border: 1px solid #ddd; border-radius: 5px;
    font-size: 16px; }
    .search-box button { padding: 12px 30px; background: #667eea; color: white; border: none;
    border-radius: 5px; cursor: pointer; }
    .filters { display: flex; gap: 10px; margin-top: 15px; }
    .filter-btn { padding: 8px 16px; background: white; border: 1px solid #ddd; border-radius:
    20px; cursor: pointer; }
    .filter-btn.active { background: #667eea; color: white; border-color: #667eea; }
    .packs-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(300px, 1fr)); gap:
    20px; margin-top: 30px; }
    .pack-card { background: white; border-radius: 10px; padding: 20px; box-shadow: 0 2px 10px
    rgba(0,0,0,0.1); cursor: pointer; transition: transform 0.2s; }
    .pack-card:hover { transform: translateY(-5px); }
    .pack-header { display: flex; justify-content: space-between; align-items: start; }
    .pack-type { background: #e9ecef; color: #495057; padding: 4px 10px; border-radius: 15px;
    font-size: 12px; }
    .pack-name { font-size: 18px; font-weight: bold; margin: 10px 0; }
    .pack-meta { color: #6c757d; font-size: 14px; }
    .pack-stats { display: flex; gap: 15px; margin-top: 15px; font-size: 14px; }

```

```

.pack-stat { display: flex; align-items: center; gap: 5px; }
.empty-state { text-align: center; padding: 60px 20px; color: #6c757d; }
.load-more { text-align: center; margin: 40px 0; }
.load-btn { padding: 12px 40px; background: white; border: 2px solid #667eea; color: #667eea; border-radius: 5px; cursor: pointer; font-size: 16px; }
</style>
</head>
<body>
  <div class="container">
    <header>
      <h1>Explore Packs</h1>
      <p>Discover and use packages created by the community</p>

      <div class="search-box">
        <input type="text" id="searchInput" placeholder="Search packages...">
        <button onclick="searchPacks()">Search</button>
      </div>

      <div class="filters">
        <div class="filter-btn active" data-type="all" onclick="setFilter('all')">All</div>
        <div class="filter-btn" data-type="npm" onclick="setFilter('npm')">NPM</div>
        <div class="filter-btn" data-type="python" onclick="setFilter('python')">Python</div>
        <div class="filter-btn" data-type="wasm" onclick="setFilter('wasm')">WASM</div>
      </div>
    </header>

    <div id="packsContainer" class="packs-grid">
      <!-- Packs will be loaded here -->
    </div>

    <div id="emptyState" class="empty-state" style="display: none;">
      <h3>No packs found</h3>
      <p>Try a different search or create your own pack!</p>
      <a href="editor.html" style="color: #667eea; text-decoration: none;">Create a Pack →</a>
    </div>

    <div class="load-more">
      <button class="load-btn" onclick="loadMore()" id="loadMoreBtn" style="display: none;">Load More</button>
    </div>
  </div>

  <script>
    let currentPage = 1;

```

```
let currentFilter = 'all';
let currentSearch = '';
let hasMore = false;

// Load packs on page load
window.onload = loadPacks;

// Enter key to search
document.getElementById('searchInput').addEventListener('keypress', (e) => {
  if (e.key === 'Enter') searchPacks();
});

async function loadPacks(reset = false) {
  if (reset) {
    currentPage = 1;
    document.getElementById('packsContainer').innerHTML = '';
    document.getElementById('loadMoreBtn').style.display = 'none';
  }

  const params = new URLSearchParams({
    page: currentPage,
    limit: 12
  });

  if (currentFilter !== 'all') params.set('type', currentFilter);
  if (currentSearch) params.set('q', currentSearch);

  const response = await fetch(`/api/Old/search?${params}`);
  const result = await response.json();

  if (result.success) {
    displayPacks(result.packs);
    hasMore = result.hasMore;

    if (hasMore) {
      document.getElementById('loadMoreBtn').style.display = 'block';
    }

    // Show/hide empty state
    if (result.packs.length === 0 && currentPage === 1) {
      document.getElementById('emptyState').style.display = 'block';
    } else {
      document.getElementById('emptyState').style.display = 'none';
    }
  }
}
```

```
}  
}
```

```
function displayPacks(packs) {  
  const container = document.getElementById('packsContainer');  
  
  packs.forEach(pack => {  
    const card = document.createElement('div');  
    card.className = 'pack-card';  
    card.onclick = () =>  
      window.open(`https://packcdn.firefly-worker.workers.dev/pack/${pack.id}`, '_blank');  
  
    card.innerHTML = `  
      <div class="pack-header">  
        <span class="pack-type">${pack.package_type.toUpperCase()}</span>  
        <span class="pack-meta">v${pack.version}</span>  
      </div>  
      <div class="pack-name">${pack.name}</div>  
      <div class="pack-stats">  
        <div class="pack-stat">👁️ ${pack.views || 0} views</div>  
        <div class="pack-stat">📦 ${pack.downloads || 0} downloads</div>  
      </div>  
      <div style="margin-top: 15px; font-size: 12px; color: #999;">  
        Created ${formatDate(pack.created_at)}  
      </div>  
    `;  
  
    container.appendChild(card);  
  });  
}  
  
function searchPacks() {  
  currentSearch = document.getElementById('searchInput').value;  
  loadPacks(true);  
}  
  
function setFilter(type) {  
  currentFilter = type;  
  document.querySelectorAll('.filter-btn').forEach(btn => {  
    btn.classList.toggle('active', btn.dataset.type === type);  
  });  
  loadPacks(true);  
}
```

```
function loadMore() {
  currentPage++;
  loadPacks(false);
}

function formatDate(dateString) {
  const date = new Date(dateString);
  const now = new Date();
  const diff = now - date;

  const minutes = Math.floor(diff / 60000);
  const hours = Math.floor(diff / 3600000);
  const days = Math.floor(diff / 86400000);

  if (minutes < 60) return `${minutes}m ago`;
  if (hours < 24) return `${hours}h ago`;
  if (days < 7) return `${days}d ago`;

  return date.toLocaleDateString();
}
</script>
</body>
</html>
```

```
public/home/index.html
<!DOCTYPE html>
<html>
<head>
  <title>PackCDN - Instant Package Publishing</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    * { margin: 0; padding: 0; box-sizing: border-box; }
    body { font-family: -apple-system, BlinkMacSystemFont, sans-serif; line-height: 1.6; color:
#333; }
```

```
.hero {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  text-align: center;
  padding: 80px 20px;
  clip-path: polygon(0 0, 100% 0, 100% 85%, 0 100%);
}
```

```
.hero h1 {
```

```
font-size: 3.5rem;  
margin-bottom: 20px;  
font-weight: 800;  
}
```

```
.hero p {  
font-size: 1.2rem;  
max-width: 600px;  
margin: 0 auto 40px;  
opacity: 0.9;  
}
```

```
.cta-buttons {  
display: flex;  
gap: 15px;  
justify-content: center;  
flex-wrap: wrap;  
}
```

```
.btn {  
padding: 15px 30px;  
border-radius: 50px;  
text-decoration: none;  
font-weight: 600;  
font-size: 1rem;  
transition: all 0.3s ease;  
}
```

```
.btn-primary {  
background: white;  
color: #667eea;  
}
```

```
.btn-secondary {  
background: transparent;  
color: white;  
border: 2px solid white;  
}
```

```
.btn:hover {  
transform: translateY(-3px);  
box-shadow: 0 10px 20px rgba(0,0,0,0.2);  
}
```

```
.features {  
  max-width: 1200px;  
  margin: -50px auto 100px;  
  padding: 0 20px;  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
  gap: 30px;  
}
```

```
.feature-card {  
  background: white;  
  padding: 40px 30px;  
  border-radius: 15px;  
  box-shadow: 0 10px 40px rgba(0,0,0,0.1);  
  text-align: center;  
}
```

```
.feature-icon {  
  font-size: 3rem;  
  margin-bottom: 20px;  
}
```

```
.feature-card h3 {  
  margin-bottom: 15px;  
  color: #2d3748;  
}
```

```
.how-it-works {  
  background: #f8f9fa;  
  padding: 100px 20px;  
  text-align: center;  
}
```

```
.steps {  
  max-width: 800px;  
  margin: 60px auto 0;  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: center;  
  gap: 40px;  
}
```

```
.step {  
  flex: 1;
```

```
    min-width: 200px;
    text-align: center;
}
```

```
.step-number {
  width: 60px;
  height: 60px;
  background: #667eea;
  color: white;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 1.5rem;
  font-weight: bold;
  margin: 0 auto 20px;
}
```

```
.terminal {
  background: #1a1a1a;
  color: #00ff9d;
  padding: 20px;
  border-radius: 10px;
  font-family: monospace;
  max-width: 800px;
  margin: 40px auto;
  text-align: left;
  overflow-x: auto;
}
```

```
.terminal .comment { color: #666; }
.terminal .command { color: white; }
```

```
footer {
  background: #2d3748;
  color: white;
  text-align: center;
  padding: 40px 20px;
  margin-top: 100px;
}
```

```
@media (max-width: 768px) {
  .hero h1 { font-size: 2.5rem; }
  .hero { padding: 60px 20px; }
```

```

    .features { margin-top: -30px; }
  }
</style>
</head>
<body>
  <section class="hero">
    <h1>🚀 PackCDN</h1>
    <p>Instantly publish and share custom packages. No registration required. Anonymous and
free.</p>
    <div class="cta-buttons">
      <a href="editor.html" class="btn btn-primary">Create a Pack</a>
      <a href="explore.html" class="btn btn-secondary">Explore Packs</a>
    </div>
  </section>

  <div class="features">
    <div class="feature-card">
      <div class="feature-icon">📦</div>
      <h3>NPM Packages</h3>
      <p>Publish JavaScript/TypeScript packages that can be imported directly in browsers or
Node.js</p>
    </div>

    <div class="feature-card">
      <div class="feature-icon">🐍</div>
      <h3>Python Modules</h3>
      <p>Share Python code as importable modules with automatic dependency analysis</p>
    </div>

    <div class="feature-card">
      <div class="feature-icon">⚡</div>
      <h3>WASM Modules</h3>
      <p>Upload and serve WebAssembly modules for high-performance applications</p>
    </div>
  </div>

  <section class="how-it-works">
    <h2 style="font-size: 2.5rem; margin-bottom: 20px;">How It Works</h2>
    <p style="max-width: 600px; margin: 0 auto 40px; color: #666;">
      Three simple steps to publish and share your code
    </p>

    <div class="steps">
      <div class="step">

```

```
<div class="step-number">1</div>
<h3>Write Code</h3>
<p>Use our built-in editor or upload existing files</p>
</div>
```

```
<div class="step">
  <div class="step-number">2</div>
  <h3>Publish</h3>
  <p>Get instant CDN URLs and installation commands</p>
</div>
```

```
<div class="step">
  <div class="step-number">3</div>
  <h3>Share & Use</h3>
  <p>Share your pack URL or install directly</p>
</div>
</div>
</section>
```

```
<div style="max-width: 1200px; margin: 0 auto; padding: 0 20px;">
  <div class="terminal">
    <span class="comment"># Create and publish a package</span><br>
    <span class="command">$ pack install my-utils
https://packcdn.firefly-worker.workers.dev/cdn/abc123</span><br><br>

    <span class="comment"># Use in JavaScript</span><br>
    <span class="command">import utils from
'https://packcdn.firefly-worker.workers.dev/cdn/abc123/index.js';</span><br><br>

    <span class="comment"># Use in HTML</span><br>
    <span class="command">&lt;script
src="https://packcdn.firefly-worker.workers.dev/cdn/abc123/index.js"&gt;&lt;/script></span>
  </div>
</div>
```

```
<footer>
  <p style="margin-bottom: 20px; font-size: 1.2rem;">PackCDN - Instant Package
Publishing</p>
  <p style="color: #a0aec0; max-width: 600px; margin: 0 auto;">
    Built with Vercel, Cloudflare Workers, and Supabase.<br>
    All users are anonymous. No registration required. Free forever.
  </p>
  <div style="margin-top: 30px;">
    <a href="editor.html" style="color: white; margin: 0 15px;">Create</a>
```

```
    <a href="explore.html" style="color: white; margin: 0 15px;">Explore</a>
    <a href="https://github.com/sussybocca/PackCDN/tree/main" style="color: white; margin: 0
15px;">GitHub</a>
  </div>
</footer>
</body>
</html>
```

public/dev-panel.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dev Panel | @random</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
      background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
      min-height: 100vh;
      padding: 20px;
    }

    .container {
      max-width: 1200px;
      margin: 0 auto;
      background: rgba(255, 255, 255, 0.95);
      backdrop-filter: blur(10px);
      border-radius: 20px;
      padding: 30px;
      box-shadow: 0 20px 60px rgba(0,0,0,0.1);
    }

    .header {
      text-align: center;
      margin-bottom: 40px;
    }
```

```
h1 {
  color: #333;
  font-size: 2.5rem;
  margin-bottom: 10px;
}

.subtitle {
  color: #666;
  font-size: 1.1rem;
}

.panel-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
  gap: 30px;
  margin-bottom: 40px;
}

.panel {
  background: white;
  border-radius: 15px;
  padding: 25px;
  box-shadow: 0 10px 30px rgba(0,0,0,0.08);
  transition: transform 0.3s ease;
}

.panel:hover {
  transform: translateY(-5px);
}

.panel-title {
  font-size: 1.3rem;
  color: #333;
  margin-bottom: 20px;
  padding-bottom: 10px;
  border-bottom: 2px solid #f0f0f0;
}

.form-group {
  margin-bottom: 20px;
}

label {
  display: block;
```

```
        margin-bottom: 8px;
        color: #555;
        font-weight: 500;
    }

    input, textarea, select {
        width: 100%;
        padding: 12px 15px;
        border: 2px solid #e0e0e0;
        border-radius: 8px;
        font-size: 1rem;
        transition: border-color 0.3s ease;
    }

    input:focus, textarea:focus, select:focus {
        outline: none;
        border-color: #f5576c;
    }

    textarea {
        min-height: 150px;
        resize: vertical;
        font-family: 'Monaco', 'Courier New', monospace;
        font-size: 0.9rem;
    }

    .btn {
        padding: 12px 25px;
        background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
        color: white;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        font-weight: 600;
        font-size: 1rem;
        transition: all 0.3s ease;
        width: 100%;
    }

    .btn:hover {
        transform: translateY(-2px);
        box-shadow: 0 10px 20px rgba(245, 87, 108, 0.3);
    }
```

```
.btn-secondary {  
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
}
```

```
.btn-loading {  
  opacity: 0.7;  
  cursor: not-allowed;  
}
```

```
.page-list {  
  list-style: none;  
}
```

```
.page-item {  
  background: #f8f9fa;  
  padding: 15px;  
  border-radius: 8px;  
  margin-bottom: 10px;  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  border-left: 4px solid #f5576c;  
}
```

```
.page-name {  
  font-weight: 600;  
  color: #333;  
}
```

```
.page-meta {  
  font-size: 0.85rem;  
  color: #666;  
  margin-top: 5px;  
}
```

```
.page-actions {  
  display: flex;  
  gap: 10px;  
}
```

```
.action-btn {  
  padding: 5px 12px;  
  background: white;  
  border: 1px solid #ddd;
```

```
border-radius: 5px;
cursor: pointer;
font-size: 0.85rem;
transition: all 0.2s ease;
}

.action-btn:hover {
  background: #f0f0f0;
}

.action-btn.edit { color: #2196f3; border-color: #2196f3; }
.action-btn.delete { color: #f44336; border-color: #f44336; }
.action-btn.view { color: #4caf50; border-color: #4caf50; }

.preview-container {
  background: #f8f9fa;
  border-radius: 10px;
  padding: 20px;
  min-height: 200px;
  border: 2px dashed #ddd;
}

.empty-state {
  text-align: center;
  color: #999;
  padding: 40px 20px;
}

.empty-state-icon {
  font-size: 3rem;
  margin-bottom: 20px;
}

.modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0,0,0,0.5);
  justify-content: center;
  align-items: center;
  z-index: 1000;
}
```

```
}
```

```
.modal.show {  
  display: flex;  
}
```

```
.modal-content {  
  background: white;  
  padding: 30px;  
  border-radius: 15px;  
  max-width: 500px;  
  width: 90%;  
  max-height: 80vh;  
  overflow-y: auto;  
}
```

```
.export-box {  
  background: #f8f9fa;  
  padding: 15px;  
  border-radius: 8px;  
  margin-top: 15px;  
}
```

```
.export-code {  
  font-family: monospace;  
  background: white;  
  padding: 10px;  
  border-radius: 5px;  
  overflow-x: auto;  
  font-size: 0.9rem;  
}
```

```
.quick-links {  
  display: flex;  
  gap: 15px;  
  flex-wrap: wrap;  
  margin-top: 30px;  
  padding-top: 30px;  
  border-top: 2px solid #f0f0f0;  
}
```

```
.quick-link {  
  padding: 10px 20px;  
  background: linear-gradient(135deg, #a8edea 0%, #fed6e3 100%);
```

```
border-radius: 8px;
text-decoration: none;
color: #333;
font-weight: 500;
transition: transform 0.3s ease;
}

.quick-link:hover {
  transform: translateY(-3px);
}

.status-message {
  padding: 10px;
  margin: 10px 0;
  border-radius: 5px;
  text-align: center;
  font-weight: 500;
}

.status-success {
  background: #d4edda;
  color: #155724;
  border: 1px solid #c3e6cb;
}

.status-error {
  background: #f8d7da;
  color: #721c24;
  border: 1px solid #f5c6cb;
}

.loading {
  text-align: center;
  padding: 20px;
}

.spinner {
  border: 3px solid #f3f3f3;
  border-top: 3px solid #f5576c;
  border-radius: 50%;
  width: 40px;
  height: 40px;
  animation: spin 1s linear infinite;
  margin: 0 auto 10px;
```

```

    }

    @keyframes spin {
      0% { transform: rotate(0deg); }
      100% { transform: rotate(360deg); }
    }

    .tags-input {
      display: flex;
      flex-wrap: wrap;
      gap: 5px;
      margin-top: 5px;
    }

    .tag {
      background: #e9ecef;
      padding: 4px 10px;
      border-radius: 15px;
      font-size: 0.85rem;
      display: flex;
      align-items: center;
      gap: 5px;
    }

    .tag-remove {
      cursor: pointer;
      color: #666;
    }

    .tag-remove:hover {
      color: #f44336;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="header">
      <h1>🚀 Developer Panel</h1>
      <p class="subtitle">Create and manage pages with Supabase API | Connected to Pack
CDN</p>
    </div>

    <div id="statusMessage" class="status-message" style="display: none;"></div>
  </div>

```

```

<div class="panel-grid">
  <!-- Create New Page -->
  <div class="panel">
    <h2 class="panel-title">Create New Page</h2>
    <div class="form-group">
      <label for="pageName">Page ID (URL)</label>
      <input type="text" id="pageName" placeholder="my-page" pattern="[a-z0-9-]+"
title="Lowercase letters, numbers, and hyphens only">
      <small style="color: #666; display: block; margin-top: 5px;">Will be accessible as
@your-page-id</small>
    </div>
    <div class="form-group">
      <label for="pageTitle">Page Title</label>
      <input type="text" id="pageTitle" placeholder="My Awesome Page">
    </div>
    <div class="form-group">
      <label for="pageDescription">Description</label>
      <input type="text" id="pageDescription" placeholder="Brief description of your
page">
    </div>
    <div class="form-group">
      <label for="pageType">Page Type</label>
      <select id="pageType">
        <option value="html">HTML Page</option>
        <option value="embed">Embed Website</option>
        <option value="markdown">Markdown</option>
        <option value="dashboard">Dashboard</option>
      </select>
    </div>
    <div class="form-group">
      <label for="pageTags">Tags (comma separated)</label>
      <input type="text" id="pageTags" placeholder="web, project, demo">
    </div>
    <div class="form-group">
      <label for="pageContent">Content</label>
      <textarea id="pageContent" placeholder="<h1>Hello World!</h1>">
<style>body { background: #f0f0f0; }</style>
<script>console.log('Page loaded!')</script>"></textarea>
    </div>
    <div class="form-group">
      <label>
        <input type="checkbox" id="isPublic" checked> Make page public
      </label>
    </div>
  </div>

```

```
<button class="btn" onclick="createPage()" id="createBtn">✨ Create Page</button>
</div>
```

```
<!-- Your Pages -->
<div class="panel">
  <h2 class="panel-title">Your Pages</h2>
  <div class="loading" id="loadingPages" style="display: none;">
    <div class="spinner"></div>
    <p>Loading pages...</p>
  </div>
  <ul class="page-list" id="pageList">
    <!-- Pages will be loaded here -->
  </ul>
  <div id="emptyState" class="empty-state">
    <div class="empty-state-icon">📄</div>
    <p>No pages created yet</p>
  </div>
</div>
```

```
<!-- Preview -->
<div class="panel">
  <h2 class="panel-title">Live Preview</h2>
  <div class="preview-container" id="previewContainer">
    <div class="empty-state">
      <div class="empty-state-icon">👁️</div>
      <p>Select a page to preview</p>
    </div>
  </div>
</div>
</div>
```

```
<!-- Quick Links -->
<div class="quick-links">
  <a href="/" class="quick-link">🏠 Home</a>
  <a href="/@cosmic" class="quick-link">🌌 Cosmic</a>
  <a href="/@neo" class="quick-link">💻 Editor</a>
  <a href="/api/Pages/Explore/explore-pages" class="quick-link">🔍 Explore Pages</a>
  <a href="#" class="quick-link" onclick="loadPages(true)">🔄 Refresh</a>
  <a href="#" class="quick-link" onclick="showAllPages()">📊 View All</a>
</div>
</div>
```

```
<!-- Modal for viewing all pages -->
<div class="modal" id="modal">
```

```
<div class="modal-content">
  <h2 id="modalTitle">All Public Pages</h2>
  <div id="modalContent">
    <!-- Content will be inserted here -->
  </div>
  <button class="btn" onclick="closeModal()" style="margin-top: 20px;">Close</button>
</div>
</div>
```

```
<script>
  // API configuration
  const API_BASE = '/api';
  let currentEditId = null;

  // Initialize
  document.addEventListener('DOMContentLoaded', () => {
    loadPages();

    // Check for URL parameters
    const urlParams = new URLSearchParams(window.location.search);
    const editId = urlParams.get('edit');
    if (editId) {
      loadPageForEdit(editId);
    }
  });

  // Show status message
  function showStatus(message, type = 'success') {
    const statusEl = document.getElementById('statusMessage');
    statusEl.textContent = message;
    statusEl.className = `status-message status-${type}`;
    statusEl.style.display = 'block';

    setTimeout(() => {
      statusEl.style.display = 'none';
    }, 5000);
  }

  // Page management
  async function loadPages(showLoading = false) {
    const pageList = document.getElementById('pageList');
    const emptyState = document.getElementById('emptyState');
    const loadingEl = document.getElementById('loadingPages');
```

```

if (showLoading) {
  loadingEl.style.display = 'block';
  pageList.innerHTML = "";
  emptyState.style.display = 'none';
}

try {
  const response = await fetch(`${API_BASE}/Pages/Explore/explore-pages`);
  if (!response.ok) throw new Error('Failed to load pages');

  const data = await response.json();
  pageList.innerHTML = "";

  if (!data.pages || data.pages.length === 0) {
    emptyState.style.display = 'block';
    return;
  }

  emptyState.style.display = 'none';

  data.pages.forEach((page) => {
    const li = document.createElement('li');
    li.className = 'page-item';
    li.innerHTML = `
      <div>
        <div class="page-name">@${page.page_id}</div>
        <div class="page-meta">
          ${page.title} • ${page.views || 0} views •
          ${new Date(page.created_at).toLocaleDateString()}
        </div>
        ${page.tags ? `<div style="margin-top: 5px;">${page.tags.map(tag => `<span
class="tag">${tag}</span>`).join("")}</div>` : ""}
      </div>
      <div class="page-actions">
        <button class="action-btn edit"
onclick="loadPageForEdit('${page.page_id}')">Edit</button>
        <button class="action-btn view"
onclick="viewPage('${page.page_id}')">View</button>
        <button class="action-btn delete"
onclick="deletePage('${page.page_id}')">Delete</button>
      </div>
    `;
    pageList.appendChild(li);
  });
}

```

```

    } catch (error) {
      console.error('Error loading pages:', error);
      pageList.innerHTML = `<li class="page-item" style="border-color: #f44336;">
        <div style="color: #f44336;">Error loading pages: ${error.message}</div>
      </li>`;
    } finally {
      loadingEl.style.display = 'none';
    }
  }
}

async function loadPageForEdit(pageId) {
  try {
    const response = await fetch(`${API_BASE}/Pages/Explore/explore-pages`);
    if (!response.ok) throw new Error('Failed to load page');

    const data = await response.json();
    const page = data.pages.find(p => p.page_id === pageId);

    if (!page) {
      showStatus('Page not found', 'error');
      return;
    }

    // Fill form
    document.getElementById('pageName').value = page.page_id;
    document.getElementById('pageTitle').value = page.title;
    document.getElementById('pageDescription').value = page.description || '';
    document.getElementById('pageType').value = page.page_type || 'html';
    document.getElementById('pageTags').value = page.tags ? page.tags.join(', ') : '';
    document.getElementById('pageContent').value = page.content || '';
    document.getElementById('isPublic').checked = page.is_public !== false;

    // Change button to update
    const btn = document.getElementById('createBtn');
    btn.textContent = '↻ Update Page';
    btn.onclick = () => updatePage(page.id);

    currentEditId = page.id;

    // Scroll to form
    document.getElementById('pageName').scrollIntoView({ behavior: 'smooth' });

    // Preview
    previewPage(page);
  }
}

```

```

        showStatus(`Loaded page @${pagelId} for editing`);
    } catch (error) {
        console.error('Error loading page:', error);
        showStatus(`Error loading page: ${error.message}`, 'error');
    }
}

async function createPage() {
    const pagelId = document.getElementById('pageName').value.trim().toLowerCase();
    const title = document.getElementById('pageTitle').value.trim();
    const description = document.getElementById('pageDescription').value.trim();
    const type = document.getElementById('pageType').value;
    const tags = document.getElementById('pageTags').value.split(',').map(t =>
t.trim()).filter(t => t);
    const content = document.getElementById('pageContent').value;
    const isPublic = document.getElementById('isPublic').checked;

    if (!pagelId) {
        showStatus('Please enter a page ID', 'error');
        return;
    }

    if (! /^[a-z0-9-]+$/.test(pagelId)) {
        showStatus('Page ID can only contain lowercase letters, numbers, and hyphens',
'error');
        return;
    }

    const btn = document.getElementById('createBtn');
    btn.textContent = 'Creating...';
    btn.classList.add('btn-loading');
    btn.disabled = true;

    try {
        const response = await fetch(`${API_BASE}/create-page`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                page_id: pagelId,
                title: title || pagelId,
                description,
            })
        });
        if (!response.ok) {
            const errorText = await response.text();
            showStatus(errorText, 'error');
            return;
        }
        const data = await response.json();
        showStatus(`Page created successfully: ${data.page_id}`, 'success');
    } catch (error) {
        console.error('Error creating page:', error);
        showStatus('Error creating page', 'error');
    }
}

```

```

        page_type: type,
        tags,
        content,
        is_public: isPublic,
        version: 1
    })
});

const data = await response.json();

if (!response.ok) {
    throw new Error(data.error || 'Failed to create page');
}

// Clear form
document.getElementById('pageName').value = "";
document.getElementById('pageTitle').value = "";
document.getElementById('pageDescription').value = "";
document.getElementById('pageTags').value = "";
document.getElementById('pageContent').value = "";

showStatus(`Page @${pageld} created successfully! Access it at @${pageld}`);

// Reload pages
loadPages();

// Preview new page
previewPage({
    page_id: pageld,
    title: title || pageld,
    page_type: type,
    content
});

} catch (error) {
    console.error('Error creating page:', error);
    showStatus(`Error: ${error.message}`, 'error');
} finally {
    btn.textContent = '✨ Create Page';
    btn.classList.remove('btn-loading');
    btn.disabled = false;
}
}

```

```
async function updatePage(pageId) {
  const title = document.getElementById('pageTitle').value.trim();
  const description = document.getElementById('pageDescription').value.trim();
  const type = document.getElementById('pageType').value;
  const tags = document.getElementById('pageTags').value.split(',').map(t =>
t.trim()).filter(t => t);
  const content = document.getElementById('pageContent').value;
  const isPublic = document.getElementById('isPublic').checked;

  const btn = document.getElementById('createBtn');
  btn.textContent = 'Updating...';
  btn.classList.add('btn-loading');
  btn.disabled = true;

  try {
    const response = await fetch(`${API_BASE}/create-page`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        id: currentEditId,
        title,
        description,
        page_type: type,
        tags,
        content,
        is_public: isPublic,
        version: 2
      })
    });
  });

  const data = await response.json();

  if (!response.ok) {
    throw new Error(data.error || 'Failed to update page');
  }

  // Reset form
  document.getElementById('pageName').value = "";
  document.getElementById('pageTitle').value = "";
  document.getElementById('pageDescription').value = "";
  document.getElementById('pageTags').value = "";
  document.getElementById('pageContent').value = "";
```

```

// Reset button
btn.textContent = '✨ Create Page';
btn.onclick = createPage;
currentEditId = null;

showStatus('Page updated successfully!');

// Reload pages
loadPages();

} catch (error) {
  console.error('Error updating page:', error);
  showStatus(`Error: ${error.message}`, 'error');
} finally {
  btn.classList.remove('btn-loading');
  btn.disabled = false;
}
}

async function deletePage(pageId) {
  if (!confirm(`Are you sure you want to delete @${pageId}? This cannot be undone!`)) {
    return;
  }

  try {
    const response = await fetch(`${API_BASE}/create-page?id=${pageId}`, {
      method: 'DELETE',
    });

    if (!response.ok) {
      throw new Error('Failed to delete page');
    }

    showStatus(`Page @${pageId} deleted`);
    loadPages();

    // Clear preview
    document.getElementById('previewContainer').innerHTML = `
      <div class="empty-state">
        <div class="empty-state-icon">👁</div>
        <p>Select a page to preview</p>
      </div>
    `;
  }
}

```

```

    } catch (error) {
      console.error('Error deleting page:', error);
      showStatus(`Error: ${error.message}`, 'error');
    }
  }

function viewPage(pageId) {
  window.location.href = `/@${pageId}`;
}

function previewPage(page) {
  const preview = document.getElementById('previewContainer');

  if (page.page_type === 'embed') {
    preview.innerHTML = `
      <div style="padding: 20px;">
        <h3>${page.title}</h3>
        <p>Embedded Website Preview</p>
        <div style="background: white; padding: 10px; border-radius: 8px; margin-top:
10px;">
          ${page.content}
        </div>
        <button class="btn btn-secondary" onclick="viewPage('${page.page_id}')"
style="margin-top: 15px;">
          Open @${page.page_id}
        </button>
      </div>
    `;
  } else {
    // Create iframe for HTML preview
    preview.innerHTML = `
      <iframe
        srcdoc="${escapeHtml(
          <!DOCTYPE html>
          <html>
          <head>
            <style>
              body {
                font-family: -apple-system, BlinkMacSystemFont, sans-serif;
                padding: 20px;
                background: white;
              }
            </style>
            .preview-header {

```

```

        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        color: white;
        padding: 20px;
        border-radius: 10px;
        margin-bottom: 20px;
    }
</style>
</head>
<body>
    <div class="preview-header">
        <h1>${page.title}</h1>
        <p>Preview of @${page.page_id}</p>
    </div>
    ${page.content || '<p>No content yet</p>'}
</body>
</html>
`))"
    style="width: 100%; height: 300px; border: none; border-radius: 8px;"
></iframe>
<div style="margin-top: 10px; text-align: center;">
    <button class="btn btn-secondary" onclick="viewPage('${page.page_id}')">
        Open @${page.page_id}
    </button>
</div>
`;
}
}

```

```

async function showAllPages() {
    try {
        const response = await fetch(`${API_BASE}/Pages/Explore/explore-pages`);
        if (!response.ok) throw new Error('Failed to load pages');

        const data = await response.json();

        let html = '<div style="max-height: 400px; overflow-y: auto;">';

        if (!data.pages || data.pages.length === 0) {
            html += '<p class="empty-state">No pages found</p>';
        } else {
            html += '<ul style="list-style: none; padding: 0;">';
            data.pages.forEach(page => {
                html += `
                    <li style="padding: 10px; border-bottom: 1px solid #eee;">

```

```

        <strong>@${page.page_id}</strong>: ${page.title}
        <div style="font-size: 0.9rem; color: #666;">
            ${page.views || 0} views • ${page.page_type} •
            ${new Date(page.created_at).toLocaleDateString()}
        </div>
    </li>
    `;
    });
    html += '</ul>';
}
html += '</div>';

document.getElementById('modalTitle').textContent = `All Pages
(${data.pages?.length || 0})`;
document.getElementById('modalContent').innerHTML = html;
document.getElementById('modal').classList.add('show');

} catch (error) {
    document.getElementById('modalContent').innerHTML = `
        <div class="status-error">Error loading pages: ${error.message}</div>
    `;
    document.getElementById('modal').classList.add('show');
}
}

function closeModal() {
    document.getElementById('modal').classList.remove('show');
}

// Utility function
function escapeHtml(unsafe) {
    return unsafe
        .replace(/&/g, "&amp;")
        .replace(/</g, "&lt;")
        .replace(/>/g, "&gt;")
        .replace(/"/g, "&quot;")
        .replace(/'/g, "&#039;");
}

// Keyboard shortcuts
document.addEventListener('keydown', (e) => {
    if (e.ctrlKey && e.key === 's') {
        e.preventDefault();
        if (currentEditId) {

```

```

        updatePage(currentEditId);
      } else {
        createPage();
      }
    }
    if (e.key === 'Escape') {
      closeModal();
    }
  });

```

```

// Auto-save draft in localStorage
const formFields = ['pageName', 'pageTitle', 'pageDescription', 'pageType', 'pageTags',
'pageContent'];

```

```

formFields.forEach(fieldId => {
  const field = document.getElementById(fieldId);
  if (field) {
    // Load saved draft
    const saved = localStorage.getItem(`draft_${fieldId}`);
    if (saved && !currentEditId) {
      if (fieldId === 'pageType') {
        field.value = saved;
      } else {
        field.value = saved;
      }
    }
  }

  // Save on change
  field.addEventListener('input', () => {
    if (!currentEditId) {
      localStorage.setItem(`draft_${fieldId}`, field.value);
    }
  });
});

```

```

// Clear draft button
document.addEventListener('keydown', (e) => {
  if (e.ctrlKey && e.key === 'd') {
    e.preventDefault();
    if (confirm('Clear all form data?')) {
      formFields.forEach(fieldId => {
        localStorage.removeItem(`draft_${fieldId}`);
        const field = document.getElementById(fieldId);

```

```
        if (field) field.value = "";
    });
    showStatus('Form cleared');
}
}
});
</script>
</body>
</html>
```

```
public/docs.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PackCDN - Interactive Documentation</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #0a0a14 0%, #1a1a2e 50%, #0f3460 100%);
      color: #ffffff;
      min-height: 100vh;
    }

    /* Animated background */
    .background-animation {
      position: fixed;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      z-index: -1;
      overflow: hidden;
    }

    .gradient-circle {
      position: absolute;
```

```
border-radius: 50%;
filter: blur(40px);
opacity: 0.3;
animation: float 20s infinite ease-in-out;
}

.circle-1 {
width: 300px;
height: 300px;
background: linear-gradient(45deg, #667eea, #764ba2);
top: 10%;
left: 10%;
animation-delay: 0s;
}

.circle-2 {
width: 400px;
height: 400px;
background: linear-gradient(45deg, #64ffda, #00b4d8);
top: 60%;
right: 10%;
animation-delay: -5s;
}

.circle-3 {
width: 250px;
height: 250px;
background: linear-gradient(45deg, #ff6b6b, #ffd93d);
bottom: 10%;
left: 20%;
animation-delay: -10s;
}

@keyframes float {
0%, 100% { transform: translateY(0) scale(1); }
33% { transform: translateY(-20px) scale(1.1); }
66% { transform: translateY(20px) scale(0.9); }
}

.nav-overlay {
position: fixed;
top: 0;
left: 0;
width: 100%;
```

```
padding: 20px;
display: flex;
justify-content: space-between;
align-items: center;
background: rgba(10, 10, 20, 0.95);
backdrop-filter: blur(10px);
border-bottom: 1px solid rgba(102, 126, 234, 0.3);
z-index: 1000;
}

.logo {
  font-size: 24px;
  font-weight: bold;
  background: linear-gradient(45deg, #667eea, #764ba2);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}

.nav-links {
  display: flex;
  gap: 30px;
}

.nav-links a {
  color: #ffffff;
  text-decoration: none;
  padding: 8px 16px;
  border-radius: 20px;
  transition: all 0.3s ease;
  font-weight: 500;
}

.nav-links a:hover {
  background: rgba(102, 126, 234, 0.3);
  transform: translateY(-2px);
}

.content {
  max-width: 1200px;
  margin: 0 auto;
  padding: 140px 40px 60px;
}

.hero-section {
```

```
text-align: center;
padding: 80px 0;
margin-bottom: 80px;
opacity: 0;
animation: fadeInUp 1s forwards;
}
```

```
.hero-title {
  font-size: 4rem;
  margin-bottom: 20px;
  background: linear-gradient(45deg, #667eea, #764ba2);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
```

```
.hero-subtitle {
  font-size: 1.5rem;
  color: #ffffff;
  margin-bottom: 40px;
  max-width: 600px;
  margin-left: auto;
  margin-right: auto;
  opacity: 0.9;
  line-height: 1.6;
}
```

```
.cta-buttons {
  display: flex;
  gap: 20px;
  justify-content: center;
}
```

```
.btn {
  padding: 15px 30px;
  border-radius: 50px;
  text-decoration: none;
  font-weight: 600;
  font-size: 1rem;
  transition: all 0.3s ease;
  cursor: pointer;
  display: inline-block;
  text-align: center;
  color: white;
}
```

```
.btn-primary {  
  background: linear-gradient(45deg, #667eea, #764ba2);  
  color: white;  
  border: none;  
}
```

```
.btn-secondary {  
  background: transparent;  
  color: #ffffff;  
  border: 2px solid #667eea;  
  background: rgba(102, 126, 234, 0.1);  
}
```

```
.btn:hover {  
  transform: translateY(-3px);  
  box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
}
```

```
.section {  
  margin: 100px 0;  
  padding: 50px;  
  background: rgba(25, 25, 35, 0.85);  
  border-radius: 20px;  
  border: 1px solid rgba(102, 126, 234, 0.3);  
  backdrop-filter: blur(10px);  
  opacity: 0;  
  transform: translateY(40px);  
  animation: fadeInUp 0.8s forwards;  
}
```

```
.section.visible {  
  opacity: 1;  
  transform: translateY(0);  
}
```

```
.section-title {  
  font-size: 2.5rem;  
  margin-bottom: 30px;  
  color: #ffffff;  
  display: flex;  
  align-items: center;  
  gap: 15px;  
}
```

```
.section-title .icon {
  font-size: 2rem;
  color: #667eea;
}

.cards-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
  gap: 30px;
  margin: 30px 0;
}

.card {
  background: rgba(35, 35, 50, 0.9);
  padding: 30px;
  border-radius: 15px;
  transition: all 0.3s ease;
  border: 1px solid rgba(102, 126, 234, 0.2);
}

.card:hover {
  transform: translateY(-10px);
  border-color: #667eea;
  box-shadow: 0 15px 30px rgba(102, 126, 234, 0.2);
}

.card-title {
  font-size: 1.5rem;
  margin-bottom: 15px;
  color: #ffffff;
  font-weight: 600;
}

.card-content {
  color: #ffffff;
  line-height: 1.6;
  opacity: 0.9;
}

.code-block {
  background: rgba(15, 15, 25, 0.95);
  padding: 25px;
  border-radius: 10px;
```

```
margin: 20px 0;
overflow-x: auto;
font-family: 'Courier New', monospace;
border-left: 4px solid #667eea;
color: #e2e8f0;
font-size: 0.95rem;
}

.code-block .highlight {
  color: #64ffda;
  font-weight: bold;
}

.terminal {
  background: rgba(10, 10, 20, 0.95);
  border: 2px solid #667eea;
  border-radius: 10px;
  overflow: hidden;
  margin: 30px 0;
}

.terminal-header {
  background: linear-gradient(45deg, #667eea, #764ba2);
  padding: 10px 20px;
  display: flex;
  align-items: center;
  gap: 10px;
}

.terminal-dot {
  width: 12px;
  height: 12px;
  border-radius: 50%;
  background: rgba(255, 255, 255, 0.7);
}

.terminal-content {
  padding: 20px;
  font-family: 'Courier New', monospace;
  color: #00ff9d;
  background: rgba(0, 0, 0, 0.5);
  line-height: 1.5;
}
```

```
.diagram-container {
  position: relative;
  height: 400px;
  margin: 40px 0;
  background: rgba(20, 20, 35, 0.8);
  border-radius: 10px;
  overflow: hidden;
  border: 1px solid rgba(102, 126, 234, 0.3);
}

.flow-node {
  position: absolute;
  width: 150px;
  padding: 20px;
  background: rgba(102, 126, 234, 0.3);
  border: 2px solid #667eea;
  border-radius: 10px;
  text-align: center;
  transition: all 0.3s ease;
  color: white;
  font-weight: 600;
  backdrop-filter: blur(5px);
  animation: floatNode 3s infinite ease-in-out;
}

.flow-node:hover {
  background: rgba(102, 126, 234, 0.5);
  transform: scale(1.05);
}

.flow-line {
  position: absolute;
  height: 2px;
  background: linear-gradient(90deg, #667eea, #764ba2);
  transform-origin: left center;
}

.feature-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
  margin-top: 30px;
}
```

```
.feature-item {
  display: flex;
  align-items: center;
  gap: 15px;
  padding: 25px;
  background: rgba(40, 40, 50, 0.85);
  border-radius: 10px;
  transition: all 0.3s ease;
  border: 1px solid rgba(102, 126, 234, 0.2);
}

.feature-item:hover {
  background: rgba(102, 126, 234, 0.3);
  transform: translateX(10px);
  border-color: #667eea;
}

.feature-icon {
  font-size: 2rem;
  color: #667eea;
}

.feature-item h3 {
  color: #ffffff;
  margin-bottom: 5px;
}

.feature-item p {
  color: #ffffff;
  opacity: 0.9;
}

.stats-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 20px;
  margin: 40px 0;
}

.stat-box {
  text-align: center;
  padding: 30px;
  background: rgba(102, 126, 234, 0.2);
  border-radius: 15px;
}
```

```
border: 1px solid rgba(102, 126, 234, 0.4);
backdrop-filter: blur(5px);
animation: pulse 2s infinite;
}

.stat-number {
  font-size: 3rem;
  font-weight: bold;
  color: #ffffff;
  margin-bottom: 10px;
}

.stat-label {
  color: #ffffff;
  font-size: 1.1rem;
  opacity: 0.9;
}

.interactive-demo {
  position: relative;
  height: 300px;
  margin: 40px 0;
  border: 2px solid rgba(102, 126, 234, 0.4);
  border-radius: 15px;
  overflow: hidden;
  background: rgba(15, 15, 25, 0.9);
  display: flex;
  align-items: center;
  justify-content: center;
}

.demo-cube {
  width: 100px;
  height: 100px;
  background: linear-gradient(45deg, #667eea, #764ba2);
  border-radius: 10px;
  animation: rotateCube 10s linear infinite;
  box-shadow: 0 10px 30px rgba(102, 126, 234, 0.5);
}

.demo-controls {
  display: flex;
  gap: 10px;
  justify-content: center;
```

```
    margin-top: 20px;
}

.demo-btn {
    padding: 12px 24px;
    background: linear-gradient(45deg, #667eea, #764ba2);
    color: white;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    transition: all 0.3s ease;
    font-weight: 600;
}

.demo-btn:hover {
    background: linear-gradient(45deg, #764ba2, #667eea);
    transform: scale(1.1);
}

footer {
    text-align: center;
    padding: 50px 40px;
    margin-top: 100px;
    border-top: 2px solid rgba(102, 126, 234, 0.3);
    color: #ffffff;
    background: rgba(10, 10, 20, 0.95);
}

footer a {
    color: #667eea;
    text-decoration: none;
    transition: all 0.3s ease;
    font-weight: 500;
}

footer a:hover {
    color: #764ba2;
    text-decoration: underline;
}

.back-to-top {
    position: fixed;
    bottom: 30px;
    right: 30px;
```

```
width: 50px;
height: 50px;
background: linear-gradient(45deg, #667eea, #764ba2);
border-radius: 50%;
display: flex;
align-items: center;
justify-content: center;
cursor: pointer;
opacity: 0;
transition: all 0.3s ease;
color: white;
font-size: 1.2rem;
z-index: 1000;
}
```

```
.back-to-top.visible {
  opacity: 1;
  transform: translateY(0);
}
```

```
.back-to-top:hover {
  background: linear-gradient(45deg, #764ba2, #667eea);
  transform: translateY(-5px) scale(1.1);
}
```

```
/* Animations */
```

```
@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(40px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

```
@keyframes floatNode {
  0%, 100% { transform: translateY(0); }
  50% { transform: translateY(-10px); }
}
```

```
@keyframes pulse {
  0%, 100% { transform: scale(1); }
```

```

    50% { transform: scale(1.05); }
  }

  @keyframes rotateCube {
    0% { transform: rotateX(0) rotateY(0) rotateZ(0); }
    100% { transform: rotateX(360deg) rotateY(360deg) rotateZ(360deg); }
  }

  @media (max-width: 768px) {
    .hero-title {
      font-size: 2.5rem;
    }

    .nav-links {
      display: none;
    }

    .content {
      padding: 140px 20px 20px;
    }

    .section {
      padding: 30px 20px;
      margin: 50px 0;
    }

    .cards-container {
      grid-template-columns: 1fr;
    }
  }
</style>
</head>
<body>
  <!-- Simple background animation -->
  <div class="background-animation">
    <div class="gradient-circle circle-1"></div>
    <div class="gradient-circle circle-2"></div>
    <div class="gradient-circle circle-3"></div>
  </div>

  <!-- Navigation -->
  <nav class="nav-overlay">
    <div class="logo">PackCDN</div>
    <div class="nav-links">

```

```
<a href="#overview">Overview</a>
<a href="#features">Features</a>
<a href="#api">API</a>
<a href="#architecture">Architecture</a>
<a href="#getting-started">Get Started</a>
<a href="selector.html">Switch View</a>
</div>
</nav>
```

```
<!-- Main content -->
```

```
<div class="content">
  <section class="hero-section">
    <h1 class="hero-title">PackCDN Documentation</h1>
    <p class="hero-subtitle">
      Instant package publishing and distribution platform with global CDN,
      built on Cloudflare Workers and Vercel Serverless Functions.
    </p>
    <div class="cta-buttons">
      <a href="#getting-started" class="btn btn-primary">Quick Start</a>
      <a href="editor.html" class="btn btn-secondary">Try Editor</a>
    </div>
  </section>
```

```

  <section id="overview" class="section">
    <h2 class="section-title"><span class="icon">📦</span> Overview</h2>
    <p class="card-content">
      PackCDN is a zero-configuration platform for instantly publishing and sharing code
      packages.
      No registration required. Anonymous and free forever.
    </p>
```

```

  <div class="stats-container">
    <div class="stat-box">
      <div class="stat-number">4</div>
      <div class="stat-label">API Endpoints</div>
    </div>
    <div class="stat-box">
      <div class="stat-number">3</div>
      <div class="stat-label">Package Types</div>
    </div>
    <div class="stat-box">
      <div class="stat-number">∞</div>
      <div class="stat-label">Global CDN</div>
    </div>
  </div>
```

```
<div class="stat-box">
  <div class="stat-number">0</div>
  <div class="stat-label">Registration</div>
</div>
</div>
</section>
```

```
<section id="features" class="section">
  <h2 class="section-title"><span class="icon"> ✨ </span> Features</h2>
```

```
<div class="feature-grid">
  <div class="feature-item">
    <span class="feature-icon"> 🚀 </span>
    <div>
      <h3>Instant Publishing</h3>
      <p>Upload code, get CDN URLs in seconds</p>
    </div>
  </div>
  <div class="feature-item">
    <span class="feature-icon"> 🌐 </span>
    <div>
      <h3>Global CDN</h3>
      <p>Cloudflare Workers edge network</p>
    </div>
  </div>
  <div class="feature-item">
    <span class="feature-icon"> 🔒 </span>
    <div>
      <h3>Anonymous</h3>
      <p>No accounts, no tracking</p>
    </div>
  </div>
  <div class="feature-item">
    <span class="feature-icon"> ⚡ </span>
    <div>
      <h3>WASM Support</h3>
      <p>Upload and serve WebAssembly</p>
    </div>
  </div>
  <div class="feature-item">
    <span class="feature-icon"> 🔍 </span>
    <div>
      <h3>Smart Analysis</h3>
      <p>Automatic dependency detection</p>
    </div>
  </div>
</div>
```

```
    </div>
  </div>
  <div class="feature-item">
    <span class="feature-icon">🔄</span>
    <div>
      <h3>Multi-format</h3>
      <p>JavaScript, Python, WASM</p>
    </div>
  </div>
</div>
</div>
</section>
```

```
<section id="api" class="section">
  <h2 class="section-title"><span class="icon">🔧</span> API Reference</h2>
```

```
  <div class="cards-container">
    <div class="card">
      <h3 class="card-title">GET /api/get-pack</h3>
      <p class="card-content">Retrieve package by ID or URL ID</p>
      <div class="code-block">
        <span class="highlight">GET</span>
        https://pack-cdn.vercel.app/api/get-pack?id={id}
      </div>
    </div>
```

```
    <div class="card">
      <h3 class="card-title">POST /api/publish</h3>
      <p class="card-content">Publish new package with CDN</p>
      <div class="code-block">
        <span class="highlight">POST</span> /api/publish<br>
        { "name": "my-package", "files": {...} }
      </div>
    </div>
```

```
    <div class="card">
      <h3 class="card-title">POST /api/analyze</h3>
      <p class="card-content">Analyze code for dependencies</p>
      <div class="code-block">
        <span class="highlight">POST</span> /api/analyze<br>
        { "code": "import React", "type": "npm" }
      </div>
    </div>
```

```
    <div class="card">
```

```
<h3 class="card-title">GET /api/search</h3>
<p class="card-content">Search public packages</p>
<div class="code-block">
  <span class="highlight">GET</span> /api/search?q=react&type=npm
</div>
</div>
</div>
```

```
<div class="terminal">
  <div class="terminal-header">
    <div class="terminal-dot"></div>
    <div class="terminal-dot"></div>
    <div class="terminal-dot"></div>
  </div>
  <div class="terminal-content">
    $ curl -X POST https://pack-cdn.vercel.app/api/publish \<br>
    -H "Content-Type: application/json" \<br>
    -d '{"name":"my-package","files":{"index.js":"..."}}'<br><br>
    {<br>
      "success": true,<br>
      "cdnUrl": "https://packcdn.../cdn/abc123",<br>
      "installCommand": "pack install my-package https://..."<br>
    }
  </div>
</div>
</section>
```

```
<section id="architecture" class="section">
  <h2 class="section-title"><span class="icon">🏗️</span> Architecture</h2>
```

```
<div class="diagram-container" id="flow-diagram">
  <!-- Flow diagram will be created with JavaScript -->
</div>
```

```
<div class="code-block">
  // Package Storage Format in Supabase<br>
  {<br>
    "url_id": "abc123xyz",<br>
    "name": "my-package",<br>
    "files": {<br>
      "index.js": "export default function() {...}",<br>
      "module.wasm": "AGFzbQEAAAAG..."<br>
    },<br>
    "pack_json": "{\"type\":\"module\",\"version\":\"1.0.0\"}"<br>
  }
```

```
    }  
  </div>  
</section>
```

```
<section id="getting-started" class="section">  
  <h2 class="section-title"><span class="icon">🚀</span> Getting Started</h2>  
  
  <div class="cards-container">  
    <div class="card">  
      <h3 class="card-title">1. Create Package</h3>  
      <p class="card-content">Use our built-in editor or upload existing files</p>  
    </div>  
  
    <div class="card">  
      <h3 class="card-title">2. Configure</h3>  
      <p class="card-content">Set package name, type, and visibility</p>  
    </div>  
  
    <div class="card">  
      <h3 class="card-title">3. Publish</h3>  
      <p class="card-content">Get instant CDN URLs and installation commands</p>  
    </div>  
  
    <div class="card">  
      <h3 class="card-title">4. Use Anywhere</h3>  
      <p class="card-content">Import directly in browser, Node.js, or other projects</p>  
    </div>  
  </div>  
  
  <div class="interactive-demo" id="demo-container">  
    <div class="demo-cube" id="demoCube"></div>  
  </div>  
  
  <div class="demo-controls">  
    <button class="demo-btn" onclick="runDemo()">Run Demo</button>  
    <button class="demo-btn" onclick="resetDemo()">Reset</button>  
  </div>  
  
  <div class="code-block">  
    // Using a published package<br>  
    import myPackage from 'https://packcdn.../cdn/abc123';<br><br>  
    // or in HTML<br>  
    &lt;script src="https://packcdn.../cdn/abc123/index.js"&gt;&lt;/script&gt;<br><br>  
    // or with curl<br>
```

```

        curl https://packcdn.../cdn/abc123/index.js
    </div>
</section>
</div>

<div class="back-to-top" id="backToTop" onclick="scrollToTop()">
    ↑
</div>

<footer>
    <p>PackCDN © 2024 | Built with Cloudflare Workers, Vercel, and Supabase</p>
    <p style="margin-top: 10px;">
        <a href="selector.html">Switch to Classic View</a> |
        <a href="home/index.html">Go to Homepage</a>
    </p>
</footer>

<script>
    // Simple scroll animations
    function initScrollAnimations() {
        const sections = document.querySelectorAll('.section');

        const observer = new IntersectionObserver((entries) => {
            entries.forEach(entry => {
                if (entry.isIntersecting) {
                    entry.target.classList.add('visible');
                }
            });
        }, {
            threshold: 0.1
        });

        sections.forEach(section => {
            observer.observe(section);
        });
    }

    // Flow diagram
    function createFlowDiagram() {
        const container = document.getElementById('flow-diagram');
        const nodes = [
            { id: 1, title: 'Editor', x: '10%', y: '20%', color: '#667eea' },
            { id: 2, title: 'API', x: '30%', y: '20%', color: '#764ba2' },
            { id: 3, title: 'Supabase', x: '50%', y: '60%', color: '#3ecf8e' },

```

```

    { id: 4, title: 'Cloudflare', x: '70%', y: '20%', color: '#f6821f' },
    { id: 5, title: 'User', x: '90%', y: '60%', color: '#1abc9c' }
  ];

  const connections = [
    { from: 1, to: 2 },
    { from: 2, to: 3 },
    { from: 3, to: 4 },
    { from: 4, to: 5 }
  ];

  // Create nodes
  nodes.forEach(node => {
    const div = document.createElement('div');
    div.className = 'flow-node';
    div.textContent = node.title;
    div.style.left = node.x;
    div.style.top = node.y;
    div.style.borderColor = node.color;
    div.style.animationDelay = `${node.id * 0.5}s`;
    container.appendChild(div);
  });
}

// Demo functions
let demoAnimation;

function runDemo() {
  const cube = document.getElementById('demoCube');
  if (demoAnimation) clearInterval(demoAnimation);

  let scale = 1;
  let growing = true;

  demoAnimation = setInterval(() => {
    if (growing) {
      scale += 0.05;
      if (scale >= 1.5) growing = false;
    } else {
      scale -= 0.05;
      if (scale <= 1) {
        clearInterval(demoAnimation);
        cube.style.transform = `scale(1)`;
        cube.style.background = 'linear-gradient(45deg, #667eea, #764ba2)';
      }
    }
  }, 100);
}

```

```

        return;
    }
}

cube.style.transform = `scale(${scale})`;
cube.style.background = scale > 1.2
    ? 'linear-gradient(45deg, #ff6b6b, #ffd93d)'
    : 'linear-gradient(45deg, #667eea, #764ba2)';
}, 50);
}

function resetDemo() {
    const cube = document.getElementById('demoCube');
    if (demoAnimation) clearInterval(demoAnimation);

    cube.style.transform = 'scale(1)';
    cube.style.background = 'linear-gradient(45deg, #667eea, #764ba2)';
}

// Back to top button
function initBackToTop() {
    const backToTop = document.getElementById('backToTop');

    window.addEventListener('scroll', () => {
        if (window.scrollY > 500) {
            backToTop.style.opacity = '1';
            backToTop.style.transform = 'translateY(0)';
        } else {
            backToTop.style.opacity = '0';
            backToTop.style.transform = 'translateY(20px)';
        }
    });
}

function scrollToTop() {
    window.scrollTo({
        top: 0,
        behavior: 'smooth'
    });
}

// Smooth scrolling for anchor links
document.querySelectorAll('a[href^="#"]').forEach(anchor => {
    anchor.addEventListener('click', function (e) {

```

```

        e.preventDefault();
        const targetId = this.getAttribute('href');
        if (targetId === '#') return;

        const targetElement = document.querySelector(targetId);
        if (targetElement) {
            targetElement.scrollIntoView({
                behavior: 'smooth'
            });
        }
    });
});

// Initialize everything
window.addEventListener('DOMContentLoaded', () => {
    initScrollAnimations();
    createFlowDiagram();
    initBackToTop();

    // Make hero section visible
    document.querySelector('.hero-section').style.opacity = '1';

    console.log("Page loaded - NO libraries, pure CSS/JS!");
});
</script>
</body>
</html>

```

```

public/embed.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web Embed | Pack CDN</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;

```

```
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
min-height: 100vh;
}
```

```
.container {
  max-width: 100%;
  height: 100vh;
  display: flex;
  flex-direction: column;
}
```

```
.header {
  background: rgba(255, 255, 255, 0.95);
  backdrop-filter: blur(10px);
  padding: 1rem;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
  display: flex;
  align-items: center;
  gap: 1rem;
}
```

```
.logo {
  font-weight: bold;
  color: #667eea;
  font-size: 1.2rem;
}
```

```
.url-bar {
  flex: 1;
  display: flex;
  gap: 0.5rem;
}
```

```
.url-input {
  flex: 1;
  padding: 0.75rem 1rem;
  border: 2px solid #e2e8f0;
  border-radius: 8px;
  font-size: 0.95rem;
  transition: all 0.3s ease;
}
```

```
.url-input:focus {
  outline: none;
}
```

```
border-color: #667eea;  
box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);  
}
```

```
.btn {  
padding: 0.75rem 1.5rem;  
background: #667eea;  
color: white;  
border: none;  
border-radius: 8px;  
cursor: pointer;  
font-weight: 500;  
transition: all 0.3s ease;  
}
```

```
.btn:hover {  
background: #5a67d8;  
transform: translateY(-1px);  
}
```

```
.btn-secondary {  
background: #718096;  
}
```

```
.btn-secondary:hover {  
background: #4a5568;  
}
```

```
.iframe-container {  
flex: 1;  
position: relative;  
background: white;  
}
```

```
iframe {  
width: 100%;  
height: 100%;  
border: none;  
}
```

```
.loading {  
position: absolute;  
top: 50%;  
left: 50%;
```

```
    transform: translate(-50%, -50%);
    text-align: center;
    color: #4a5568;
}

.spinner {
    border: 4px solid #f3f3f3;
    border-top: 4px solid #667eea;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    animation: spin 1s linear infinite;
    margin: 0 auto 1rem;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

.controls {
    display: flex;
    gap: 0.5rem;
}

.control-btn {
    padding: 0.5rem 1rem;
    background: white;
    border: 2px solid #e2e8f0;
    border-radius: 6px;
    cursor: pointer;
    transition: all 0.3s ease;
}

.control-btn:hover {
    border-color: #667eea;
    background: #f7fafc;
}

.notification {
    position: fixed;
    bottom: 20px;
    right: 20px;
    background: white;
```

```

padding: 1rem;
border-radius: 8px;
box-shadow: 0 4px 15px rgba(0,0,0,0.2);
transform: translateY(100px);
opacity: 0;
transition: all 0.3s ease;
}

.notification.show {
  transform: translateY(0);
  opacity: 1;
}
</style>
</head>
<body>
  <div class="container">
    <div class="header">
      <div class="logo"><img alt="Pack CDN logo" data-bbox="325 403 348 421"/> Pack CDN</div>
      <div class="url-bar">
        <input type="text" class="url-input" id="urlInput" placeholder="Enter website URL or search term...">
        <button class="btn" onclick="loadWebsite()">Go</button>
        <button class="btn btn-secondary" onclick="goHome()">Home</button>
      </div>
      <div class="controls">
        <button class="control-btn" onclick="goBack()">←</button>
        <button class="control-btn" onclick="goForward()">→</button>
        <button class="control-btn" onclick="reload()">↻</button>
      </div>
    </div>

    <div class="iframe-container">
      <div class="loading" id="loading">
        <div class="spinner"></div>
        <p>Loading website...</p>
      </div>
      <iframe id="websiteFrame" sandbox="allow-same-origin allow-scripts allow-forms allow-popups allow-modals" allow="camera; microphone; geolocation"></iframe>
    </div>
  </div>

  <div class="notification" id="notification"></div>

</script>

```

```
// Get URL from query parameters
const params = new URLSearchParams(window.location.search);
let currentUrl = params.get('url') || params.get('q');

// If coming from @anything pattern
if (window.location.pathname.startsWith('/@embed/')) {
    const encodedUrl = window.location.pathname.split('/@embed/')[1];
    currentUrl = decodeURIComponent(encodedUrl);
}

const urlInput = document.getElementById('urlInput');
const websiteFrame = document.getElementById('websiteFrame');
const loading = document.getElementById('loading');
const notification = document.getElementById('notification');

// Initialize
if (currentUrl) {
    urlInput.value = currentUrl;
    loadWebsite(currentUrl);
}

function loadWebsite(url = null) {
    const inputUrl = url || urlInput.value.trim();
    if (!inputUrl) return;

    loading.style.display = 'block';
    websiteFrame.style.opacity = '0';

    // Process the URL
    let targetUrl = inputUrl;

    // If it's a search term (no dots, not a URL)
    if (!inputUrl.includes('.') && !inputUrl.includes('/:/')) {
        targetUrl = `https://duckduckgo.com/html/?q=${encodeURIComponent(inputUrl)}`;
    }
    // If missing protocol
    else if (!inputUrl.startsWith('http://') && !inputUrl.startsWith('https://')) {
        targetUrl = 'https://' + inputUrl;
    }

    // Update URL input
    urlInput.value = targetUrl;

    // Update browser URL without reload
```

```
const embedPath = `/@embed/${encodeURIComponent(targetUrl)}`;
window.history.pushState({}, "", embedPath);

// Load in iframe
websiteFrame.src = targetUrl;

// Handle iframe load
websiteFrame.onload = () => {
  loading.style.display = 'none';
  websiteFrame.style.opacity = '1';
  showNotification(`Loaded: ${new URL(targetUrl).hostname}`);
};

websiteFrame.onerror = () => {
  loading.style.display = 'none';
  showNotification('Failed to load website. Trying proxy...', 'error');
  // Fallback to proxy
  websiteFrame.src = `/api/proxy?url=${encodeURIComponent(targetUrl)}`;
};
}

function goHome() {
  window.location.href = '/';
}

function goBack() {
  if (websiteFrame.contentWindow.history.length > 1) {
    websiteFrame.contentWindow.history.back();
  }
}

function goForward() {
  websiteFrame.contentWindow.history.forward();
}

function reload() {
  websiteFrame.contentWindow.location.reload();
}

function showNotification(message, type = 'info') {
  notification.textContent = message;
  notification.className = `notification show ${type}`;
  setTimeout(() => {
    notification.classList.remove('show');
  }, 3000);
}
```

```

    }, 3000);
}

// Handle Enter key in input
urlInput.addEventListener('keypress', (e) => {
    if (e.key === 'Enter') loadWebsite();
});

// Handle browser navigation
window.addEventListener('popstate', () => {
    const params = new URLSearchParams(window.location.search);
    const url = params.get('url');
    if (url) {
        urlInput.value = url;
        loadWebsite(url);
    }
});
</script>
</body>
</html>

public/editor.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pack Editor - WebAssembly Support</title>
    <link rel="stylesheet" data-name="vs/editor/editor.main"
href="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/editor/editor.main.min.c
ss">
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #1e1e1e;
            color: #d4d4d4;
            display: flex;
            flex-direction: column;

```

```
    height: 100vh;
    overflow: hidden;
}

header {
    background-color: #252526;
    padding: 15px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px solid #3e3e42;
}

.logo {
    font-size: 1.5rem;
    font-weight: bold;
    color: #569cd6;
}

.controls {
    display: flex;
    gap: 10px;
}

button {
    background-color: #0e639c;
    color: white;
    border: none;
    padding: 8px 16px;
    border-radius: 4px;
    cursor: pointer;
    font-weight: 500;
    transition: background-color 0.2s;
}

button:hover {
    background-color: #1177bb;
}

button.secondary {
    background-color: #3e3e42;
}

button.secondary:hover {
```

```
        background-color: #4a4a4f;
    }

    button.success {
        background-color: #388a34;
    }

    button.success:hover {
        background-color: #3fa33a;
    }

    button.disabled {
        background-color: #3e3e42;
        cursor: not-allowed;
        opacity: 0.6;
    }

    .container {
        display: flex;
        flex: 1;
        overflow: hidden;
    }

    .sidebar {
        width: 280px;
        background-color: #252526;
        border-right: 1px solid #3e3e42;
        padding: 20px;
        overflow-y: auto;
    }

    .sidebar h3 {
        margin-bottom: 15px;
        color: #cccccc;
        font-size: 1rem;
        font-weight: 500;
    }

    .file-list {
        list-style-type: none;
        margin-bottom: 30px;
        max-height: 200px;
        overflow-y: auto;
    }
```

```
.file-list li {  
  padding: 8px 12px;  
  border-radius: 4px;  
  margin-bottom: 5px;  
  cursor: pointer;  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
}
```

```
.file-list li:hover {  
  background-color: #2a2d2e;  
}
```

```
.file-list li.active {  
  background-color: #094771;  
}
```

```
.file-icon {  
  color: #c586c0;  
  font-size: 0.9rem;  
}
```

```
.json-file .file-icon {  
  color: #f5d76e;  
}
```

```
.js-file .file-icon {  
  color: #4ec9b0;  
}
```

```
.wasm-file .file-icon {  
  color: #d16969;  
}
```

```
.file-actions {  
  display: flex;  
  gap: 5px;  
  opacity: 0;  
  transition: opacity 0.2s;  
}
```

```
.file-list li:hover .file-actions {
```

```
    opacity: 1;
}

.file-action-btn {
    background: none;
    border: none;
    color: #858585;
    padding: 2px 5px;
    font-size: 0.8rem;
}

.file-action-btn:hover {
    color: #d4d4d4;
}

.wasm-section {
    margin-top: 20px;
    border-top: 1px solid #3e3e42;
    padding-top: 20px;
}

.wasm-info {
    background-color: #2a2d2e;
    padding: 10px;
    border-radius: 4px;
    margin-bottom: 15px;
    font-size: 0.85rem;
}

.wasm-info.warning {
    background-color: #5c2d0c;
    border-left: 3px solid #f5d76e;
}

.wasm-info.success {
    background-color: #2d5c0c;
    border-left: 3px solid #3fa33a;
}

.upload-area {
    border: 2px dashed #3e3e42;
    border-radius: 8px;
    padding: 20px;
    text-align: center;
```

```
    cursor: pointer;
    transition: border-color 0.2s;
    margin-bottom: 15px;
}

.upload-area:hover {
    border-color: #0e639c;
}

.upload-area.dragover {
    border-color: #569cd6;
    background-color: rgba(86, 156, 214, 0.1);
}

.upload-icon {
    font-size: 2rem;
    color: #569cd6;
    margin-bottom: 10px;
}

.upload-area p {
    margin-bottom: 5px;
}

.upload-hint {
    font-size: 0.8rem;
    color: #858585;
}

.wasm-list {
    list-style-type: none;
    margin-top: 10px;
    max-height: 150px;
    overflow-y: auto;
}

.wasm-list li {
    padding: 8px 12px;
    background-color: #2a2d2e;
    border-radius: 4px;
    margin-bottom: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
```

```
    font-size: 0.9rem;
}

.wasm-size {
    color: #858585;
    font-size: 0.8rem;
}

.editor-container {
    flex: 1;
    display: flex;
    flex-direction: column;
    overflow: hidden;
}

.editor-tabs {
    display: flex;
    background-color: #2d2d30;
    border-bottom: 1px solid #3e3e42;
    overflow-x: auto;
}

.tab {
    padding: 10px 20px;
    border-right: 1px solid #3e3e42;
    cursor: pointer;
    white-space: nowrap;
    display: flex;
    align-items: center;
    gap: 8px;
}

.tab:hover {
    background-color: #2a2d2e;
}

.tab.active {
    background-color: #1e1e1e;
}

.close-tab {
    margin-left: 5px;
    opacity: 0.7;
}
```

```
.close-tab:hover {
  opacity: 1;
}

#editor {
  flex: 1;
  overflow: hidden;
}

.config-panel {
  margin-top: 20px;
}

.config-item {
  margin-bottom: 15px;
}

label {
  display: block;
  margin-bottom: 5px;
  font-size: 0.9rem;
  color: #cccccc;
}

input[type="text"], select {
  width: 100%;
  padding: 8px 12px;
  background-color: #3e3e42;
  border: 1px solid #565656;
  border-radius: 4px;
  color: #d4d4d4;
}

input[type="text"]:focus, select:focus {
  outline: none;
  border-color: #007acc;
}

.checkbox-group {
  display: flex;
  align-items: center;
  gap: 8px;
  margin-top: 5px;
```

```
}

input[type="checkbox"] {
  accent-color: #0e639c;
}

.publish-modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.8);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1000;
  display: none;
}

.modal-content {
  background-color: #252526;
  border-radius: 8px;
  padding: 25px;
  width: 90%;
  max-width: 500px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.5);
}

.modal-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.modal-header h2 {
  color: #569cd6;
  font-size: 1.5rem;
}

.close-modal {
  font-size: 1.8rem;
  cursor: pointer;
}
```

```
        color: #cccccc;
    }

    .close-modal:hover {
        color: white;
    }

    .publish-result {
        background-color: #1e1e1e;
        padding: 15px;
        border-radius: 4px;
        margin-top: 20px;
        font-family: monospace;
        font-size: 0.9rem;
        white-space: pre-wrap;
        word-break: break-all;
        display: none;
    }

    .result-item {
        margin-bottom: 10px;
    }

    .result-label {
        color: #9cdcfe;
        font-weight: bold;
    }

    .result-value {
        color: #ce9178;
    }

    .result-value a {
        color: #569cd6;
        text-decoration: none;
    }

    .result-value a:hover {
        text-decoration: underline;
    }

    .loading {
        display: none;
        text-align: center;
```

```
    margin: 20px 0;
    color: #569cd6;
}

.spinner {
    border: 3px solid rgba(86, 156, 214, 0.3);
    border-radius: 50%;
    border-top: 3px solid #569cd6;
    width: 30px;
    height: 30px;
    animation: spin 1s linear infinite;
    margin: 0 auto 10px;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

.notification {
    position: fixed;
    bottom: 20px;
    right: 20px;
    padding: 12px 20px;
    background-color: #0c7b93;
    color: white;
    border-radius: 4px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
    z-index: 1001;
    transform: translateY(100px);
    opacity: 0;
    transition: transform 0.3s, opacity 0.3s;
}

.notification.show {
    transform: translateY(0);
    opacity: 1;
}

.notification.error {
    background-color: #a12c2c;
}

.notification.success {
```

```

        background-color: #388a34;
    }

    footer {
        padding: 10px 20px;
        background-color: #252526;
        border-top: 1px solid #3e3e42;
        font-size: 0.8rem;
        color: #858585;
        text-align: center;
    }
</style>
</head>
<body>
    <header>
        <div class="logo">Pack Editor</div>
        <div class="controls">
            <button id="newFileBtn" class="secondary">New File</button>
            <button id="saveBtn">Save</button>
            <button id="publishBtn" class="success">Publish Pack</button>
        </div>
    </header>

    <div class="container">
        <div class="sidebar">
            <h3>Files</h3>
            <ul class="file-list" id="fileList">
                <!-- Files will be added here dynamically -->
            </ul>

            <div class="wasm-section">
                <h3>WebAssembly Files</h3>

                <div class="wasm-info warning" id="wasmInfo">
                    <strong>Important:</strong> WASM files cannot be edited directly. You can only
upload valid WebAssembly binary files (.wasm). These will be served from the CDN.
                </div>

                <div class="upload-area" id="uploadArea">
                    <div class="upload-icon">⬆️</div>
                    <p>Drag & drop WASM files here</p>
                    <p class="upload-hint">or click to browse</p>
                    <p class="upload-hint">Only .wasm files accepted</p>
                </div>
            </div>
        </div>
    </div>

```

```

        <ul class="wasm-list" id="wasmList">
            <!-- WASM files will be listed here -->
        </ul>
    </div>

    <div class="config-panel">
        <h3>Pack Configuration</h3>

        <div class="config-item">
            <label for="packName">Pack Name</label>
            <input type="text" id="packName" placeholder="my-awesome-pack"
value="my-awesome-pack">
        </div>

        <div class="config-item">
            <label for="packType">Package Type</label>
            <select id="packType">
                <option value="module">Module</option>
                <option value="library">Library</option>
                <option value="template">Template</option>
                <option value="plugin">Plugin</option>
            </select>
        </div>

        <div class="config-item">
            <label>Visibility</label>
            <div class="checkbox-group">
                <input type="checkbox" id="isPublic" checked>
                <label for="isPublic">Public Package</label>
            </div>
        </div>
    </div>
</div>

<div class="editor-container">
    <div class="editor-tabs" id="editorTabs">
        <!-- Editor tabs will be added here dynamically -->
    </div>
    <div id="editor"></div>
</div>

<div class="publish-modal" id="publishModal">

```

```

<div class="modal-content">
  <div class="modal-header">
    <h2>Publish Pack</h2>
    <div class="close-modal" id="closeModal">&times;</div>
  </div>

  <p>Ready to publish your pack? This will upload all files and generate a unique URL for
your package.</p>

  <div id="validationResults">
    <!-- Validation results will appear here -->
  </div>

  <div class="loading" id="publishLoading">
    <div class="spinner"></div>
    <p>Publishing your pack...</p>
  </div>

  <div class="publish-result" id="publishResult">
    <!-- Publish results will be displayed here -->
  </div>

  <div style="display: flex; gap: 10px; margin-top: 20px;">
    <button id="confirmPublishBtn" class="success" style="flex: 1;">Publish
Now</button>
    <button id="cancelPublishBtn" class="secondary">Cancel</button>
  </div>
</div>

<div class="notification" id="notification">Message</div>

<footer>
  Pack Editor - Edit and publish packages with WebAssembly support
</footer>

<!-- Monaco Editor loader -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs/loader.min.js"></script>
<script>
  // Monaco editor configuration
  require.config({ paths: { vs:
'https://cdnjs.cloudflare.com/ajax/libs/monaco-editor/0.44.0/min/vs' } });

```

```

// Initialize app state
const appState = {
  files: {
    'package.json': {
      name: 'package.json',
      content: JSON.stringify({
        name: 'my-awesome-pack',
        version: '1.0.0',
        type: 'module',
        description: 'My awesome package with WASM support',
        main: 'index.js',
        author: '',
        license: 'MIT'
      }, null, 2),
      language: 'json',
      editable: true
    },
    'index.js': {
      name: 'index.js',
      content: `// Main entry point for your package

export function helloWorld() {
  console.log('Hello from my awesome pack!');
  return 'Hello World!';
}

// Add your code here
export const version = '1.0.0';

// Example utility function
export function sum(a, b) {
  return a + b;
}`,
      language: 'javascript',
      editable: true
    },
    'README.md': {
      name: 'README.md',
      content: `# My Awesome Pack

```

This is a sample package created with Pack Editor.

Installation

```
\\\`bash
pack install my-awesome-pack
\\\`
```

Usage

```
\\\`javascript
import { helloWorld } from 'my-awesome-pack';

helloWorld(); // Outputs: Hello from my awesome pack!
\\\`
```

API

helloWorld()

Prints a greeting message and returns "Hello World!"

sum(a, b)

Returns the sum of two numbers

License

MIT

```
`;
    language: 'markdown',
    editable: true
  },
  wasmFiles: {},
  activeFile: 'package.json',
  editor: null,
  packName: 'my-awesome-pack',
  packType: 'module',
  isPublic: true
};

// DOM Elements
const fileList = document.getElementById('fileList');
const editorTabs = document.getElementById('editorTabs');
const editorContainer = document.getElementById('editor');
const packNameInput = document.getElementById('packName');
const packTypeSelect = document.getElementById('packType');
const isPublicCheckbox = document.getElementById('isPublic');
const publishModal = document.getElementById('publishModal');
```

```
const closeModal = document.getElementById('closeModal');
const cancelPublishBtn = document.getElementById('cancelPublishBtn');
const confirmPublishBtn = document.getElementById('confirmPublishBtn');
const publishResult = document.getElementById('publishResult');
const publishLoading = document.getElementById('publishLoading');
const notification = document.getElementById('notification');
const uploadArea = document.getElementById('uploadArea');
const wasmList = document.getElementById('wasmList');
const wasmInfo = document.getElementById('wasmInfo');
const validationResults = document.getElementById('validationResults');
```

```
// File input for WASM uploads (hidden)
const fileInput = document.createElement('input');
fileInput.type = 'file';
fileInput.accept = '.wasm';
fileInput.multiple = true;
fileInput.style.display = 'none';
document.body.appendChild(fileInput);
```

```
// Initialize Monaco Editor
require(['vs/editor/editor.main'], function() {
  // Create editor instance
  appState.editor = monaco.editor.create(editorContainer, {
    value: appState.files['package.json'].content,
    language: 'json',
    theme: 'vs-dark',
    minimap: { enabled: true },
    scrollBeyondLastLine: false,
    automaticLayout: true,
    fontSize: 14,
    lineNumbers: 'on',
    roundedSelection: false,
    scrollbar: {
      vertical: 'auto',
      horizontal: 'auto'
    },
  },
  readOnly: false
});
```

```
// Update app state when editor content changes
appState.editor.onDidChangeModelContent(() => {
  const activeFile = appState.activeFile;
  if (activeFile && appState.files[activeFile]) {
    appState.files[activeFile].content = appState.editor.getValue();
  }
});
```

```

// If this is package.json, update the pack name if changed
if (activeFile === 'package.json') {
  try {
    const packageJson = JSON.parse(appState.editor.getValue());
    if (packageJson.name && packageJson.name !== appState.packName) {
      appState.packName = packageJson.name;
      packNameInput.value = packageJson.name;
    }
    if (packageJson.type && packageJson.type !== appState.packType) {
      appState.packType = packageJson.type;
      packTypeSelect.value = packageJson.type;
    }
  } catch (e) {
    // Invalid JSON, ignore
  }
}
});

// Initialize UI
renderFileList();
renderEditorTabs();
setupEventListeners();
});

// Render the file list in the sidebar
function renderFileList() {
  fileList.innerHTML = "";

  Object.keys(appState.files).forEach(fileName => {
    const file = appState.files[fileName];
    const li = document.createElement('li');
    li.className = `file-item ${fileName === appState.activeFile ? 'active' : ''}
${file.language}-file`;

    const fileExt = fileName.split('.').pop();
    const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📄' : fileExt === 'md' ? '📖' : '📄';

    li.innerHTML = `
    <div>
      <span class="file-icon">${icon}</span>
      <span>${fileName}</span>
    </div>
  `;
  });
}

```

```

        <div class="file-actions">
            <button class="file-action-btn" data-file="${fileName}"
title="Rename"><img alt="pencil icon" data-bbox="248 128 268 148"/></button>
            ${fileName !== 'package.json' && fileName !== 'index.js' ?
            `<button class="file-action-btn" data-file="${fileName}"
title="Delete"><img alt="trash icon" data-bbox="238 183 258 203"/></button>` : ""}
        </div>
    `;

```

```

    li.addEventListener('click', (e) => {
        if (!e.target.closest('.file-action-btn')) {
            switchFile(fileName);
        }
    });

```

```

    fileList.appendChild(li);
});

```

```

// Add event listeners for file actions
document.querySelectorAll('.file-action-btn').forEach(btn => {
    btn.addEventListener('click', (e) => {
        e.stopPropagation();
        const fileName = btn.dataset.file;

        if (btn.textContent.includes('<img alt="pencil icon" data-bbox="438 533 458 553"/>')) {
            renameFile(fileName);
        } else if (btn.textContent.includes('<img alt="trash icon" data-bbox="488 568 508 588"/>')) {
            deleteFile(fileName);
        }
    });
});
}

```

```

// Render WASM file list
function renderWasmList() {
    wasmList.innerHTML = "";

```

```

    if (Object.keys(appState.wasmFiles).length === 0) {
        const li = document.createElement('li');
        li.textContent = 'No WASM files uploaded';
        li.style.color = '#858585';
        li.style.fontStyle = 'italic';
        wasmList.appendChild(li);
        wasmInfo.className = 'wasm-info warning';
    }
}

```

wasmlInfo.innerHTML = 'Important: WASM files cannot be edited directly. You can only upload valid WebAssembly binary files (.wasm). These will be served from the CDN.';

```
    } else {
      Object.keys(appState.wasmFiles).forEach(fileName => {
        const file = appState.wasmFiles[fileName];
        const li = document.createElement('li');

        li.innerHTML = `
          <div>
            <span class="file-icon">⚡ </span>
            <span>${fileName}</span>
          </div>
          <div>
            <span class="wasm-size">${formatFileSize(file.size)}</span>
            <button class="file-action-btn" data-wasm="${fileName}"
title="Remove">🗑️ </button>
          </div>
        `;

        wasmList.appendChild(li);
      });

      wasmlInfo.className = 'wasm-info success';
      wasmlInfo.innerHTML = `<strong>✓ Ready:</strong>
${Object.keys(appState.wasmFiles).length} WASM file(s) uploaded and validated.`;

      // Add event listeners for remove buttons
      document.querySelectorAll('[data-wasm]').forEach(btn => {
        btn.addEventListener('click', (e) => {
          const fileName = btn.dataset.wasm;
          deleteWasmFile(fileName);
        });
      });
    }
  }

  // Render editor tabs
  function renderEditorTabs() {
    editorTabs.innerHTML = "";

    Object.keys(appState.files).forEach(fileName => {
      const tab = document.createElement('div');
      tab.className = `tab ${fileName === appState.activeFile ? 'active' : ''}`;
```

```

    tab.dataset.file = fileName;

    const fileExt = fileName.split('.').pop();
    const icon = fileExt === 'js' ? '📄' : fileExt === 'json' ? '📋' : fileExt === 'md' ? '📖' : '📄';

    tab.innerHTML = `
      <span class="file-icon">${icon}</span>
      <span>${fileName}</span>
      ${fileName !== 'package.json' && fileName !== 'index.js' ?
        '<span class="close-tab" data-file="' + fileName + '">✕</span>' : ''}
    `;

    tab.addEventListener('click', (e) => {
      if (!e.target.classList.contains('close-tab')) {
        switchFile(fileName);
      }
    });

    editorTabs.appendChild(tab);
  });

  // Add event listeners for close tab buttons
  document.querySelectorAll('.close-tab').forEach(btn => {
    btn.addEventListener('click', (e) => {
      e.stopPropagation();
      const fileName = btn.dataset.file;
      deleteFile(fileName);
    });
  });
}

// Switch to a different file
function switchFile(fileName) {
  if (!appState.files[fileName]) return;

  appState.activeFile = fileName;
  const file = appState.files[fileName];

  // Update editor content and language
  appState.editor.setValue(file.content);
  monaco.editor.setModelLanguage(appState.editor.getModel(), file.language);

  // Update UI
  renderFileList();
}

```

```

    renderEditorTabs();
  }

  // Create a new file
  function createNewFile() {
    const fileName = prompt('Enter file name (with extension, e.g., utils.js):', 'newfile.js');
    if (!fileName) return;

    if (appState.files[fileName] || appState.wasmFiles[fileName]) {
      showNotification('File already exists!', 'error');
      return;
    }

    const fileExt = fileName.split('.').pop();
    let language = 'plaintext';
    let content = '';

    switch (fileExt) {
      case 'js':
        language = 'javascript';
        content = `// ${fileName}\n\n`;
        break;
      case 'json':
        language = 'json';
        content = '{\n  \n}\n';
        break;
      case 'md':
        language = 'markdown';
        content = `# ${fileName}\n\n`;
        break;
      case 'html':
        language = 'html';
        content = `<!DOCTYPE html>\n<html>\n<head>\n
<title></title>\n</head>\n<body>\n  \n</body>\n</html>\n`;
        break;
      case 'css':
        language = 'css';
        content = `/* ${fileName} */\n\n`;
        break;
    }

    appState.files[fileName] = {
      name: fileName,
      content,
    }
  }

```

```

        language,
        editable: true
    };

    switchFile(fileName);
    showNotification(`Created ${fileName}`, 'success');
}

// Rename a file
function renameFile(oldName) {
    const newName = prompt('Enter new file name:', oldName);
    if (!newName || newName === oldName) return;

    if (appState.files[newName] || appState.wasmFiles[newName]) {
        showNotification('File already exists!', 'error');
        return;
    }

    appState.files[newName] = {
        ...appState.files[oldName],
        name: newName
    };

    delete appState.files[oldName];

    if (appState.activeFile === oldName) {
        appState.activeFile = newName;
    }

    renderFileList();
    renderEditorTabs();
    showNotification(`Renamed ${oldName} to ${newName}`, 'success');
}

// Delete a file
function deleteFile(fileName) {
    if (fileName === 'package.json' || fileName === 'index.js') {
        showNotification('Cannot delete required files!', 'error');
        return;
    }

    if (confirm(`Are you sure you want to delete ${fileName}?`)) {
        delete appState.files[fileName];
    }
}

```

```

    if (appState.activeFile === fileName) {
      const remainingFiles = Object.keys(appState.files);
      if (remainingFiles.length > 0) {
        switchFile(remainingFiles[0]);
      }
    }

    renderFileList();
    renderEditorTabs();
    showNotification(`Deleted ${fileName}`, 'success');
  }
}

// Handle WASM file upload
async function handleWasmUpload(files) {
  for (const file of files) {
    if (!file.name.endsWith('.wasm')) {
      showNotification(`Skipped ${file.name}: Not a .wasm file`, 'error');
      continue;
    }

    // Validate WASM file
    const isValid = await validateWasmFile(file);
    if (!isValid) {
      showNotification(`Skipped ${file.name}: Invalid WebAssembly file`, 'error');
      continue;
    }

    // Read file as ArrayBuffer
    const reader = new FileReader();
    reader.onload = async (e) => {
      const arrayBuffer = e.target.result;

      // Store WASM file
      appState.wasmFiles[file.name] = {
        name: file.name,
        size: file.size,
        data: arrayBuffer
      };

      renderWasmList();
      showNotification(`Uploaded ${file.name} (${formatFileSize(file.size)})`, 'success');
    };
  }
}

```

```

        reader.readAsArrayBuffer(file);
    }
}

// Validate a WASM file
async function validateWasmFile(file) {
    try {
        // Basic WASM validation - check magic number
        const arrayBuffer = await file.slice(0, 8).arrayBuffer();
        const header = new Uint8Array(arrayBuffer);

        // WASM magic number: \0asm
        const wasmMagic = [0x00, 0x61, 0x73, 0x6D];

        for (let i = 0; i < 4; i++) {
            if (header[i] !== wasmMagic[i]) {
                return false;
            }
        }

        return true;
    } catch (error) {
        console.error('WASM validation error:', error);
        return false;
    }
}

// Delete a WASM file
function deleteWasmFile(fileName) {
    if (confirm(`Are you sure you want to remove ${fileName}?`)) {
        delete appState.wasmFiles[fileName];
        renderWasmList();
        showNotification(`Removed ${fileName}`, 'success');
    }
}

// Format file size
function formatFileSize(bytes) {
    if (bytes === 0) return '0 Bytes';
    const k = 1024;
    const sizes = ['Bytes', 'KB', 'MB', 'GB'];
    const i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i];
}

```

```

// Show notification
function showNotification(message, type = 'info') {
  notification.textContent = message;
  notification.className = `notification ${type}`;
  notification.classList.add('show');

  setTimeout(() => {
    notification.classList.remove('show');
  }, 3000);
}

// Validate package before publishing
function validatePackage() {
  const errors = [];
  const warnings = [];

  // Check package.json
  try {
    const packageJson = JSON.parse(appState.files['package.json'].content);

    if (!packageJson.name || packageJson.name.trim() === '') {
      errors.push('Package name is required in package.json');
    }

    if (!packageJson.version) {
      warnings.push('Consider adding a version field to package.json');
    }

    if (!packageJson.main) {
      warnings.push('Consider specifying a main entry point in package.json');
    }
  } catch (e) {
    errors.push('Invalid JSON in package.json');
  }

  // Check if index.js exists
  if (!appState.files['index.js']) {
    warnings.push('index.js is recommended as the main entry point');
  }

  // Check for invalid file names
  Object.keys(appState.files).forEach(fileName => {
    if (fileName.includes(' ')) {

```

```
        warnings.push(`File "${fileName}" contains spaces, consider using hyphens or
underscores`);
    }
});
```

```
    return { errors, warnings };
}
```

```
// Publish package to API - FIXED FOR API EXPECTATIONS
```

```
async function publishPackage() {
```

```
    // Validate package first
```

```
    const validation = validatePackage();
```

```
    if (validation.errors.length > 0) {
```

```
        showNotification('Cannot publish: ' + validation.errors[0], 'error');
```

```
        return;
```

```
    }
```

```
    // Show loading state
```

```
    publishLoading.style.display = 'block';
```

```
    validationResults.innerHTML = "";
```

```
    try {
```

```
        // Prepare package.json
```

```
        const packageJson = JSON.parse(appState.files['package.json'].content);
```

```
        packageJson.name = appState.packName;
```

```
        packageJson.type = appState.packType;
```

```
        // Prepare all files for upload - SIMPLE OBJECT AS API EXPECTS
```

```
        const files = {};
```

```
        // Add regular files - just the content as string
```

```
        Object.keys(appState.files).forEach(fileName => {
```

```
            files[fileName] = appState.files[fileName].content;
```

```
        });
```

```
        // Add WASM files - convert ArrayBuffer to base64 string
```

```
        Object.keys(appState.wasmFiles).forEach(fileName => {
```

```
            const wasmFile = appState.wasmFiles[fileName];
```

```
            // Convert ArrayBuffer to base64 string
```

```
            const bytes = new Uint8Array(wasmFile.data);
```

```
            let binary = "";
```

```
            for (let i = 0; i < bytes.byteLength; i++) {
```

```
                binary += String.fromCharCode(bytes[i]);
```

```

    }
    files[fileName] = btoa(binary); // Just the base64 string, not an object
  });

  // Prepare request body - EXACTLY AS API EXPECTS
  const requestBody = {
    name: appState.packName,
    packJson: JSON.stringify(packageJson), // Already a string in your API
    files: files, // Simple object: {filename: contentString}
    isPublic: appState.isPublic
  };

  console.log('Sending to API:', {
    name: requestBody.name,
    packJson: requestBody.packJson.substring(0, 100) + '...',
    fileCount: Object.keys(requestBody.files).length,
    fileNames: Object.keys(requestBody.files)
  });

  // Call publish API
  const response = await fetch('/api/publish.js', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestBody)
  });

  const responseText = await response.text();
  console.log('API Response:', responseText);

  if (!response.ok) {
    throw new Error(`HTTP ${response.status}: ${response.statusText}. Response:
    ${responseText}`);
  }

  const result = JSON.parse(responseText);

  // Hide loading, show results
  publishLoading.style.display = 'none';
  publishResult.style.display = 'block';

  // Display results
  publishResult.innerHTML = `

```

```

<div class="result-item">
  <span class="result-label">✓ Published successfully!</span>
</div>
<div class="result-item">
  <span class="result-label">Pack ID:</span>
  <span class="result-value">${result.packId}</span>
</div>
<div class="result-item">
  <span class="result-label">CDN URL:</span>
  <span class="result-value"><a href="${result.cdnUrl}"
target="_blank">${result.cdnUrl}</a></span>
</div>
<div class="result-item">
  <span class="result-label">Worker URL:</span>
  <span class="result-value"><a href="${result.workerUrl}"
target="_blank">${result.workerUrl}</a></span>
</div>
<div class="result-item">
  <span class="result-label">Install Command:</span>
  <span class="result-value">${result.installCommand}</span>
</div>
  ${result.encryptedKey ? `
<div class="result-item">
  <span class="result-label">Encryption Key:</span>
  <span class="result-value">${result.encryptedKey}</span>
</div>
<div class="result-item">
  <span class="result-label">⚠ Save this key! It won't be shown again.</span>
</div>
  ` : ""}
  <div class="result-item">
    <button id="exploreBtn" class="success" style="width: 100%; margin-top:
10px;">Explore Package</button>
  </div>
  `;

```

```

// Add event listener for explore button
document.getElementById('exploreBtn').addEventListener('click', () => {
  window.location.href = 'explore.html';
});

} catch (error) {
  publishLoading.style.display = 'none';
  showNotification(`Publish failed: ${error.message}`, 'error');
}

```

```

        console.error('Publish error:', error);
    }
}

// Setup event listeners
function setupEventListeners() {
    // Pack configuration
    packNameInput.addEventListener('input', (e) => {
        appState.packName = e.target.value;
    });

    packTypeSelect.addEventListener('change', (e) => {
        appState.packType = e.target.value;
    });

    isPublicCheckbox.addEventListener('change', (e) => {
        appState.isPublic = e.target.checked;
    });

    // New file button
    document.getElementById('newFileBtn').addEventListener('click', createNewFile);

    // Save button
    document.getElementById('saveBtn').addEventListener('click', () => {
        showNotification('All changes saved locally', 'success');
    });

    // Publish button
    document.getElementById('publishBtn').addEventListener('click', () => {
        // Validate before showing modal
        const validation = validatePackage();

        validationResults.innerHTML = "";

        if (validation.errors.length > 0) {
            validationResults.innerHTML = `
                <div style="background-color: #5c2d0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
                    <strong>Errors:</strong>
                    <ul style="margin: 5px 0 0 20px;">
                        ${validation.errors.map(err => `<li>${err}</li>`).join("")}
                    </ul>
                </div>
            `;
        }
    });
}

```

```

    } else if (validation.warnings.length > 0) {
      validationResults.innerHTML = `
        <div style="background-color: #5c5c0c; padding: 10px; border-radius: 4px;
margin: 10px 0;">
          <strong>Warnings:</strong>
          <ul style="margin: 5px 0 0 20px;">
            ${validation.warnings.map(warn => `<li>${warn}</li>`).join("")}
          </ul>
        </div>
      `;
    }

    publishModal.style.display = 'flex';
    publishResult.style.display = 'none';
    publishLoading.style.display = 'none';
  });

  // Modal controls
  closeModal.addEventListener('click', () => {
    publishModal.style.display = 'none';
  });

  cancelPublishBtn.addEventListener('click', () => {
    publishModal.style.display = 'none';
  });

  confirmPublishBtn.addEventListener('click', publishPackage);

  // WASM file upload
  uploadArea.addEventListener('click', () => {
    fileInput.click();
  });

  fileInput.addEventListener('change', (e) => {
    handleWasmUpload(Array.from(e.target.files));
    fileInput.value = "";
  });

  // Drag and drop for WASM files
  uploadArea.addEventListener('dragover', (e) => {
    e.preventDefault();
    uploadArea.classList.add('dragover');
  });

```

```

        uploadArea.addEventListener('dragleave', () => {
            uploadArea.classList.remove('dragover');
        });

        uploadArea.addEventListener('drop', (e) => {
            e.preventDefault();
            uploadArea.classList.remove('dragover');

            const files = Array.from(e.dataTransfer.files);
            handleWasmUpload(files);
        });
    }

    // Initialize WASM list
    renderWasmList();

    // Close modal when clicking outside
    window.addEventListener('click', (e) => {
        if (e.target === publishModal) {
            publishModal.style.display = 'none';
        }
    });
</script>
</body>
</html>

public/selector.html
<!-- selector.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PackCDN - Select Experience</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
            background: linear-gradient(135deg, #0a0a14 0%, #1a1a2e 100%);

```

```
    min-height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    color: #f0f0f0;
}

.container {
    max-width: 1200px;
    padding: 40px;
    text-align: center;
}

.logo {
    font-size: 4rem;
    font-weight: 800;
    margin-bottom: 20px;
    background: linear-gradient(45deg, #667eea, #764ba2);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}

.tagline {
    font-size: 1.2rem;
    color: #a0aec0;
    margin-bottom: 60px;
    max-width: 600px;
    margin-left: auto;
    margin-right: auto;
}

.selector-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 30px;
    margin: 40px 0;
}

.experience-card {
    background: rgba(30, 30, 40, 0.8);
    border-radius: 20px;
    padding: 40px 30px;
    cursor: pointer;
    transition: all 0.3s ease;
```

```
border: 2px solid transparent;
backdrop-filter: blur(10px);
}

.experience-card:hover {
  transform: translateY(-10px) scale(1.02);
  border-color: #667eea;
  box-shadow: 0 20px 40px rgba(102, 126, 234, 0.3);
}

.experience-card.highlight {
  background: linear-gradient(135deg, rgba(102, 126, 234, 0.1), rgba(118, 75, 162, 0.1));
  border-color: #667eea;
}

.card-icon {
  font-size: 3rem;
  margin-bottom: 20px;
}

.card-title {
  font-size: 1.8rem;
  margin-bottom: 15px;
  color: #f0f0f0;
}

.card-description {
  color: #a0aec0;
  line-height: 1.6;
  margin-bottom: 20px;
}

.card-features {
  list-style: none;
  text-align: left;
  margin-top: 20px;
}

.card-features li {
  padding: 8px 0;
  color: #a0aec0;
  position: relative;
  padding-left: 25px;
}
```

```
.card-features li:before {  
  content: "✓";  
  position: absolute;  
  left: 0;  
  color: #667eea;  
  font-weight: bold;  
}
```

```
.select-btn {  
  margin-top: 20px;  
  padding: 12px 30px;  
  background: #667eea;  
  color: white;  
  border: none;  
  border-radius: 50px;  
  font-size: 1rem;  
  font-weight: 600;  
  cursor: pointer;  
  transition: all 0.3s ease;  
  width: 100%;  
}
```

```
.select-btn:hover {  
  background: #764ba2;  
  transform: translateY(-2px);  
}
```

```
.quick-links {  
  margin-top: 50px;  
  padding-top: 30px;  
  border-top: 1px solid rgba(102, 126, 234, 0.3);  
}
```

```
.quick-links h3 {  
  margin-bottom: 20px;  
  color: #667eea;  
}
```

```
.link-buttons {  
  display: flex;  
  gap: 15px;  
  justify-content: center;  
  flex-wrap: wrap;
```

```
}

.link-btn {
  padding: 10px 25px;
  background: rgba(102, 126, 234, 0.1);
  color: #667eea;
  border: 1px solid #667eea;
  border-radius: 50px;
  text-decoration: none;
  transition: all 0.3s ease;
}

.link-btn:hover {
  background: #667eea;
  color: white;
}

footer {
  margin-top: 40px;
  color: #666;
  font-size: 0.9rem;
}

@media (max-width: 768px) {
  .container {
    padding: 20px;
  }

  .logo {
    font-size: 2.5rem;
  }

  .selector-grid {
    grid-template-columns: 1fr;
  }
}

.floating {
  animation: floating 6s ease-in-out infinite;
}

@keyframes floating {
  0%, 100% { transform: translateY(0); }
  50% { transform: translateY(-20px); }
```

```

    }
  </style>
</head>
<body>
  <div class="container">
    <h1 class="logo floating">PackCDN</h1>
    <p class="tagline">Choose your experience - Instant package publishing and distribution platform</p>

    <div class="selector-grid">
      <div class="experience-card highlight">
        <div class="card-icon">✨</div>
        <h2 class="card-title">Immersive Docs</h2>
        <p class="card-description">
          Interactive 3D documentation with animated examples, live demos,
          and visual architecture diagrams.
        </p>
        <ul class="card-features">
          <li>Three.js powered 3D background</li>
          <li>GSAP animations throughout</li>
          <li>Interactive code demos</li>
          <li>Visual flow diagrams</li>
          <li>Smooth scroll animations</li>
        </ul>
        <button class="select-btn" onclick="window.location.href='Docs/docs.html'">
          Enter Immersive Experience
        </button>
      </div>

      <div class="experience-card">
        <div class="card-icon">🚀</div>
        <h2 class="card-title">Classic Website</h2>
        <p class="card-description">
          Fast, lightweight landing page with quick access to all features
          and straightforward navigation.
        </p>
        <ul class="card-features">
          <li>Lightweight & fast loading</li>
          <li>Traditional navigation</li>
          <li>Quick access to features</li>
          <li>Mobile optimized</li>
          <li>Simple and clean</li>
        </ul>
        <button class="select-btn" onclick="window.location.href='home/index.html'">

```

```

        Go to Classic Website
      </button>
    </div>
  </div>

  <div class="quick-links">
    <h3>Quick Access</h3>
    <div class="link-buttons">
      <a href="Editor/editor.html" class="link-btn">Create Package</a>
      <a href="Explore/explore.html" class="link-btn">Explore Packs</a>
      <a href="/api/search" class="link-btn">API Search</a>
      <a href="https://github.com/sussybocca/PackCDN/tree/main"
class="link-btn">GitHub</a>
    </div>
  </div>

  <footer>
    PackCDN © 2024 | Anonymous Package Publishing | Built with Cloudflare Workers &
Vercel
  </footer>
</div>

<script>
  // Add floating animation to cards
  document.querySelectorAll('.experience-card').forEach((card, index) => {
    card.style.animationDelay = `${index * 0.2}s`;
    card.classList.add('floating');
  });

  // Add click effects
  document.querySelectorAll('.experience-card').forEach(card => {
    card.addEventListener('click', function(e) {
      if (!e.target.classList.contains('select-btn')) {
        const btn = this.querySelector('.select-btn');
        if (btn) {
          btn.style.transform = 'scale(0.95)';
          setTimeout(() => {
            btn.style.transform = "";
            btn.click();
          }, 150);
        }
      }
    });
  });
};

```

```

// Add keyboard navigation
document.addEventListener('keydown', (e) => {
  if (e.key === '1') {
    window.location.href = 'docs.html';
  } else if (e.key === '2') {
    window.location.href = 'home/index.html';
  } else if (e.key === 'e') {
    window.location.href = 'editor.html';
  }
});

// Add welcome message
console.log(`
🚀 Welcome to PackCDN!
=====
Quick Navigation:
• Press 1 for Immersive Docs
• Press 2 for Classic Website
• Press E for Editor

Choose your experience above!
`);
</script>
</body>
</html>

```

Cloudflare_Workers

Cloudflare_Workers/cdn_worker.js

```

// Cloudflare Worker - packcdn.firefly-worker.workers.dev
export default {
  async fetch(request, env) {
    const url = new URL(request.url);
    const path = url.pathname;
    const hostname = request.headers.get('host') || 'packcdn.firefly-worker.workers.dev';

    // Get the origin from the request
    const requestOrigin = request.headers.get('Origin');

    // Allowed origins - your Vercel site and localhost for development
    const allowedOrigins = [
      'https://pack-cdn.vercel.app',
      'http://localhost:3000',

```

```

    'http://localhost:5173',
    'http://127.0.0.1:3000',
    'http://127.0.0.1:5173',
    `https://${hostname}`,
    `http://${hostname}`
  ];

  // Determine if the origin is allowed
  let corsOrigin = '*';
  if (requestOrigin && allowedOrigins.includes(requestOrigin)) {
    corsOrigin = requestOrigin;
  }

  // CORS headers
  const corsHeaders = {
    'Access-Control-Allow-Origin': corsOrigin,
    'Access-Control-Allow-Methods': 'GET, POST, PUT, DELETE, OPTIONS',
    'Access-Control-Allow-Headers': 'Content-Type, Authorization, Accept, Origin,
X-Requested-With',
    'Access-Control-Max-Age': '86400',
    'Vary': 'Origin',
  };

  // Handle CORS preflight
  if (request.method === 'OPTIONS') {
    return new Response(null, {
      status: 204,
      headers: corsHeaders
    });
  }

  // Route handlers
  let response;
  try {
    if (path.startsWith('/cdn/')) {
      response = await handleCDN(request, path, hostname);
    } else if (path.startsWith('/pack/')) {
      response = await handlePackInfo(request, path, hostname);
    } else if (path.startsWith('/search')) {
      response = await handleSearch(request, url, hostname);
    } else if (path === '/') {
      response = handleHome(hostname);
    } else {
      response = new Response('Not Found', {

```

```

        status: 404,
        headers: { 'Content-Type': 'text/plain' }
    });
}
} catch (error) {
    console.error('Worker error:', error);
    response = new Response('Internal Server Error: ' + error.message, {
        status: 500,
        headers: { 'Content-Type': 'text/plain' }
    });
}

// Add CORS headers to response
const headers = new Headers(response.headers);
for (const [key, value] of Object.entries(corsHeaders)) {
    if (key !== 'Vary') {
        headers.set(key, value);
    }
}

return new Response(response.body, {
    status: response.status,
    statusText: response.statusText,
    headers: headers
});
}
};

// Handle CDN file serving
async function handleCDN(request, path, hostname) {
    const segments = path.split('/');
    const packId = segments[2];
    const filePath = segments.slice(3).join('/') || 'index.js';

    if (!packId) {
        return new Response('Missing pack ID', {
            status: 400,
            headers: { 'Content-Type': 'text/plain' }
        });
    }

    try {
        console.log(`Fetching pack: ${packId}, file: ${filePath}`);
    }

```

```

// Fetch pack from the get-pack API
const apiResponse = await fetch(
  `https://pack-cdn.vercel.app/api/get-pack?id=${encodeURIComponent(packId)}`,
  {
    headers: {
      'User-Agent': 'PackCDN-Worker/1.0',
      'Accept': 'application/json',
      'Origin': `https://${hostname}`
    }
  }
);

console.log(`API response status: ${apiResponse.status}`);

if (!apiResponse.ok) {
  const errorText = await apiResponse.text();
  console.error(`API error: ${apiResponse.status} - ${errorText}`);
  return new Response(`Pack not found: ${packId} (API: ${apiResponse.status})`, {
    status: 404,
    headers: { 'Content-Type': 'text/plain' }
  });
}

const result = await apiResponse.json();
console.log(`API result success: ${result.success}`);

if (!result.success || !result.pack) {
  return new Response(`Invalid pack data: ${packId} - ${result.error || 'No pack data'}`, {
    status: 404,
    headers: { 'Content-Type': 'text/plain' }
  });
}

const pack = result.pack;
console.log(`Pack found: ${pack.name || pack.id}, files count: ${pack.files ?
Object.keys(pack.files).length : 0}`);

// Get files from pack data
if (!pack.files || typeof pack.files !== 'object') {
  console.error('No files object in pack:', pack);
  return new Response('No files in pack', {
    status: 404,
    headers: { 'Content-Type': 'text/plain' }
  });
}

```

```

    }

    // Find the requested file
    let fileContent = pack.files[filePath];

    // If file not found, try index.js as default
    if (!fileContent && filePath === 'index.js') {
        console.log('File not found, trying fallbacks...');
        // Try common entry points
        fileContent = pack.files['main.js'] ||
            pack.files['index.mjs'] ||
            pack.files['app.js'];

        // If still not found, try to find any .js file
        if (!fileContent) {
            const jsFiles = Object.entries(pack.files).find(([name, content]) =>
                name.endsWith('.js') || name.endsWith('.mjs') || name.endsWith('.cjs')
            );
            if (jsFiles) {
                fileContent = jsFiles[1];
                console.log(`Found fallback JS file: ${jsFiles[0]}`);
            }
        }

        // Last resort: first file
        if (!fileContent) {
            const firstFile = Object.values(pack.files)[0];
            if (firstFile) {
                fileContent = firstFile;
                console.log('Using first file as fallback');
            }
        }
    }

    if (!fileContent) {
        console.error(`File not found in pack: ${filePath}, available files:`, Object.keys(pack.files));
        return new Response(`File not found: ${filePath}. Available files:
${Object.keys(pack.files).join(', ')}`, {
            status: 404,
            headers: { 'Content-Type': 'text/plain' }
        });
    }

    console.log(`Serving file: ${filePath}, content length: ${fileContent.length}`);

```

```

// Determine content type
let contentType = 'text/plain';
const ext = filePath.split('.').pop().toLowerCase();

if (ext === 'js' || ext === 'mjs' || ext === 'cjs') {
  contentType = 'application/javascript';
} else if (ext === 'py') {
  contentType = 'text/x-python';
} else if (ext === 'wasm') {
  contentType = 'application/wasm';
  // Check if WASM is base64 encoded
  if (typeof fileContent === 'string' && fileContent.startsWith('data:application/wasm;base64,'))
{
    const base64Data = fileContent.split(',')[1];
    fileContent = Uint8Array.from(atob(base64Data), c => c.charCodeAt(0));
  }
} else if (ext === 'json') {
  contentType = 'application/json';
} else if (ext === 'html') {
  contentType = 'text/html';
} else if (ext === 'css') {
  contentType = 'text/css';
} else if (ext === 'md') {
  contentType = 'text/markdown';
}

const responseHeaders = {
  'Content-Type': contentType,
  'Cache-Control': 'public, max-age=86400',
  'X-Pack-ID': packId,
  'X-File-Path': filePath
};

// Add content length for binary data
if (fileContent instanceof Uint8Array) {
  responseHeaders['Content-Length'] = fileContent.length.toString();
}

return new Response(fileContent, {
  headers: responseHeaders
});

} catch (error) {

```

```

    console.error('CDN error:', error);
    return new Response(`CDN Error: ${error.message}`, {
      status: 500,
      headers: { 'Content-Type': 'text/plain' }
    });
  }
}

// Handle search - proxy to your search API
async function handleSearch(request, url, hostname) {
  const searchParams = url.searchParams;

  // Build query for your search API
  const searchUrl = new URL('https://pack-cdn.vercel.app/api/search.js');
  searchParams.forEach((value, key) => {
    searchUrl.searchParams.set(key, value);
  });

  try {
    const response = await fetch(searchUrl, {
      headers: {
        'User-Agent': 'PackCDN-Worker/1.0',
        'Accept': 'application/json',
        'Origin': `https://${hostname}`
      }
    });

    if (!response.ok) {
      return new Response(`Search API error: ${response.status}`, {
        status: response.status,
        headers: { 'Content-Type': 'application/json' }
      });
    }

    const data = await response.json();

    return new Response(JSON.stringify(data), {
      headers: {
        'Content-Type': 'application/json',
        'Cache-Control': 'public, max-age=60'
      }
    });
  } catch (error) {

```

```

    console.error('Search error:', error);
    return new Response(
      JSON.stringify({
        success: false,
        error: 'Search failed',
        packs: []
      }),
      {
        status: 500,
        headers: { 'Content-Type': 'application/json' }
      }
    );
  }
}

// Handle pack information page
async function handlePackInfo(request, path, hostname) {
  const packId = path.split('/')[2];

  if (!packId) {
    return renderNotFound('Missing pack ID', hostname);
  }

  try {
    // Fetch pack data
    const response = await fetch(
      `https://pack-cdn.vercel.app/api/get-pack?id=${encodeURIComponent(packId)}`,
      {
        headers: {
          'Origin': `https://${hostname}`
        }
      }
    );

    if (!response.ok) {
      return renderNotFound(`Pack ${packId} not found`, hostname);
    }

    const result = await response.json();

    if (!result.success || !result.pack) {
      return renderNotFound(result.error || 'Pack not found', hostname);
    }
  }
}

```

```

const pack = result.pack;

// Generate HTML page
const html = `
<!DOCTYPE html>
<html>
<head>
  <title>${pack.name || packId} - PackCDN</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body { font-family: sans-serif; max-width: 800px; margin: 0 auto; padding: 20px; }
    .header { background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); color: white;
padding: 2rem; border-radius: 10px; margin-bottom: 2rem; }
    .badge { background: #4CAF50; color: white; padding: 4px 12px; border-radius: 20px;
font-size: 14px; margin-right: 8px; }
    .code { background: #1a1a1a; color: #00ff9d; padding: 1rem; border-radius: 5px;
font-family: monospace; margin: 1rem 0; }
    .files { margin-top: 2rem; }
    .file-item { padding: 10px; border-bottom: 1px solid #eee; }
  </style>
</head>
<body>
  <div class="header">
    <h1>${pack.name || 'Unnamed Pack'}</h1>
    <p>${pack.version ? `v${pack.version}` : ''} • ${pack.package_type || 'npm'} •
${pack.downloads || 0} downloads</p>
    <span class="badge">${pack.is_public ? 'Public' : 'Private'}</span>
  </div>

  <div class="code">
    $ pack install ${pack.name || packId} https://${hostname}/cdn/${packId}
  </div>

  <h3>Usage in JavaScript</h3>
  <div class="code">
    import pkg from 'https://${hostname}/cdn/${packId}';<br>
    // or<br>
    const pkg = await import('https://${hostname}/cdn/${packId}');
  </div>

  <h3>Usage in HTML</h3>
  <div class="code">
    &lt;script src="https://${hostname}/cdn/${packId}/index.js"&gt;&lt;/script>
  </div>

```

```

    ${pack.pack_json && pack.pack_json.description ? `
    <h3>Description</h3>
    <p>${pack.pack_json.description}</p>
    ` : ""}

    <div class="files">
      <h3>Files (${pack.files ? Object.keys(pack.files).length : 0})</h3>
      ${pack.files ? Object.keys(pack.files).map(file => `
        <div class="file-item">
          <strong>${file}</strong>
          ${file === 'index.js' || file === 'main.js' ? '<span class="badge">Entry</span>' : ''}
        </div>
      `).join("") : '<p>No files</p>'}
    </div>

    <div style="margin-top: 2rem; color: #666;">
      <p>Pack ID: ${pack.id}</p>
      <p>Created: ${new Date(pack.created_at).toLocaleDateString()}</p>
      <p>CDN URL: <a href="/cdn/${packId}">/cdn/${packId}</a></p>
    </div>
  </body>
</html>
`;

return new Response(html, {
  headers: { 'Content-Type': 'text/html' }
});

} catch (error) {
  console.error('Pack info error:', error);
  return renderNotFound('Error loading pack', hostname);
}
}

function handleHome(hostname) {
  const html = `<!DOCTYPE html>
<html>
<head>
  <title>PackCDN - Custom Package CDN</title>
  <style>
    body { font-family: sans-serif; text-align: center; padding: 50px; }
    h1 { color: #667eea; font-size: 3em; }

```

```

.cta { background: #667eea; color: white; padding: 15px 30px; border-radius: 5px;
text-decoration: none; display: inline-block; margin: 20px; }
.code { background: #1a1a1a; color: #00ff9d; padding: 1rem; border-radius: 5px; font-family:
monospace; margin: 20px auto; max-width: 600px; }
</style>
</head>
<body>
<h1>🚀 PackCDN</h1>
<p>Create, publish, and serve custom packages instantly</p>

<div class="code">
  # Install any pack<br>
  $ pack install my-package https://{hostname}/cdn/package-id
</div>

<a href="https://pack-cdn.vercel.app/editor.html" class="cta">Create a Pack</a>
<a href="https://pack-cdn.vercel.app/explore.html" class="cta">Explore Packs</a>

<div style="margin-top: 50px;">
  <h3>How it works:</h3>
  <p>1. Create your package using the editor</p>
  <p>2. Publish to get a CDN URL</p>
  <p>3. Use anywhere with the CDN link</p>
</div>
</body>
</html>`;

```

```

return new Response(html, {
  headers: { 'Content-Type': 'text/html' }
});
}

```

```

function renderNotFound(message = 'Pack not found', hostname) {
  const html = `<!DOCTYPE html>
<html>
<body style="text-align: center; padding: 50px; font-family: sans-serif;">
  <h1>${message}</h1>
  <a href="https://{hostname}">Back to PackCDN</a>
</body>
</html>`;
  return new Response(html, {
    status: 404,
    headers: { 'Content-Type': 'text/html' }
  });
}

```

```
}
```

```
Cloudflare_Configuration
```

```
name = "packcdn"
```

```
main = "Clouflare_Workers/cdn_worker.js"
```

```
compatibility_date = "2024-01-01"
```

```
compatibility_flags = ["nodejs_compat"]
```

```
[env.production]
```

```
workers_dev = true
```

```
[env.development]
```

```
workers_dev = true
```

```
[dev]
```

```
port = 8787
```

```
ip = "127.0.0.1"
```

```
[vars]
```

```
API_BASE_URL = "https://pack-cdn.vercel.app/api"
```

```
DEFAULT_ORIGIN = ""
```

```
# CORS configuration
```

```
[[cors]]
```

```
origins = [
```

```
    "https://pack-cdn.vercel.app",
```

```
    "http://localhost:3000",
```

```
    "http://localhost:5173",
```

```
    "http://127.0.0.1:3000",
```

```
    "http://127.0.0.1:5173"
```

```
]
```

```
methods = ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
```

```
headers = [
```

```
    "Content-Type",
```

```
    "Authorization",
```

```
    "Accept",
```

```
    "Origin",
```

```
    "X-Requested-With",
```

```
    "Access-Control-Allow-Origin"
```

```
]
```

```
max_age = 86400
```

Vercel_Configuration

```
{
  "rewrites": [
    {
      "source": "/",
      "destination": "/selector.html"
    },
    {
      "source": "/home",
      "destination": "/home/index.html"
    },
    {
      "source": "/docs",
      "destination": "/docs.html"
    },
    {
      "source": "/editor",
      "destination": "/editor.html"
    },
    {
      "source": "/explore",
      "destination": "/explore.html"
    },
    {
      "source": "/config.json",
      "destination": "/config.json"
    },
    {
      "source": "/api/random-urls",
      "destination": "/api/random-urls"
    },
    {
      "source": "/api/wildcard-redirect",
      "destination": "/api/wildcard-redirect"
    },
    {
      "source": "/api/admin-auth",
      "destination": "/api/admin-auth"
    },
    {
      "source": "/api/embed-website",
      "destination": "/api/embed-website"
    },
    {

```

```
"source": "/api/create-page",
"destination": "/api/create-page"
},
{
  "source": "/api/Pages/Explore/explore-pages",
  "destination": "/api/Pages/Explore/explore-pages"
},
{
  "source": "/@quantum",
  "destination": "/selector.html"
},
{
  "source": "/@cosmic",
  "destination": "/home/index.html"
},
{
  "source": "/@digital",
  "destination": "/docs.html"
},
{
  "source": "/@neo",
  "destination": "/editor.html"
},
{
  "source": "/@virtual",
  "destination": "/explore.html"
},
{
  "source": "/@synth",
  "destination": "/config.json"
},
{
  "source": "/@stellar",
  "destination": "/Docs/docs.html"
},
{
  "source": "/@cyber",
  "destination": "/Editor/editor.html"
},
{
  "source": "/@orbital",
  "destination": "/Explore/explore.html"
},
{

```

```
"source": "/@dragon",
"destination": "/Login/login.html"
},
{
  "source": "/@phoenix",
  "destination": "/api/admin-auth"
},
{
  "source": "/@homecore",
  "destination": "/Docs/Pages/Home/index.html"
},
{
  "source": "/@editornetwork",
  "destination": "/Docs/Pages/Home/Pages/Editor/editor.html"
},
{
  "source": "/@explorervortex",
  "destination": "/Docs/Pages/Home/Pages/Explore/explore.html"
},
{
  "source": "/@homestudio",
  "destination": "/home/editor.html"
},
{
  "source": "/@homediscovery",
  "destination": "/home/explore.html"
},
{
  "source": "/@random",
  "destination": "/api/wildcard-redirect"
},
{
  "source": "/@embed/:url*",
  "destination": "/embed.html?url=:url*"
},
{
  "source": "/@:username",
  "destination": "/api/wildcard-redirect"
},
{
  "source": "/selector.html",
  "destination": "/@quantum"
},
{

```

```
"source": "/home/index.html",
"destination": "/@cosmic"
},
{
  "source": "/docs.html",
  "destination": "/@digital"
},
{
  "source": "/editor.html",
  "destination": "/@neo"
},
{
  "source": "/explore.html",
  "destination": "/@virtual"
},
{
  "source": "/config.json",
  "destination": "/@synth"
},
{
  "source": "/Docs/docs.html",
  "destination": "/@stellar"
},
{
  "source": "/Editor/editor.html",
  "destination": "/@cyber"
},
{
  "source": "/Explore/explore.html",
  "destination": "/@orbital"
},
{
  "source": "/Login/login.html",
  "destination": "/@dragon"
},
{
  "source": "/Private/admin.html",
  "destination": "/api/admin-auth"
},
{
  "source": "/Docs/Pages/Home/index.html",
  "destination": "/@homecore"
},
{
```

```
    "source": "/Docs/Pages/Home/Pages/Editor/editor.html",
    "destination": "/@editornetwork"
  },
  {
    "source": "/Docs/Pages/Home/Pages/Explore/explore.html",
    "destination": "/@explorervortex"
  },
  {
    "source": "/home/editor.html",
    "destination": "/@homestudio"
  },
  {
    "source": "/home/explore.html",
    "destination": "/@homediscovery"
  },
  {
    "source": "/api/wildcard-redirect",
    "destination": "/@random"
  },
  {
    "source": "/(.*)",
    "destination": "/api/wildcard-redirect"
  }
],
"redirects": [
  {
    "source": "/portal",
    "destination": "/@quantum",
    "permanent": true
  },
  {
    "source": "/gateway",
    "destination": "/@cosmic",
    "permanent": true
  },
  {
    "source": "/library",
    "destination": "/@digital",
    "permanent": true
  },
  {
    "source": "/workshop",
    "destination": "/@neo",
    "permanent": true
  }
]
```

```
},
{
  "source": "/discover",
  "destination": "/@virtual",
  "permanent": true
},
{
  "source": "/settings",
  "destination": "/@synth",
  "permanent": true
},
{
  "source": "/dev",
  "destination": "/@random",
  "permanent": true
},
{
  "source": "/create",
  "destination": "/@random",
  "permanent": true
},
{
  "source": "/explore-old",
  "destination": "/api/Pages/Explore/explore-pages",
  "permanent": true
}
],
"headers": [
  {
    "source": "/config.json",
    "headers": [
      {
        "key": "Content-Type",
        "value": "application/json"
      },
      {
        "key": "Access-Control-Allow-Origin",
        "value": "*"
      }
    ]
  },
  {
    "source": "/embed.html",
    "headers": [
```

```

        {
            "key": "Content-Security-Policy",
            "value": "default-src 'self' https: data: 'unsafe-inline' 'unsafe-eval' blob:; frame-src *;"
        },
        {
            "key": "X-Frame-Options",
            "value": "ALLOWALL"
        }
    ]
},
{
    "source": "/api/Pages/Explore/explore-pages",
    "headers": [
        {
            "key": "Access-Control-Allow-Origin",
            "value": "*"
        },
        {
            "key": "Cache-Control",
            "value": "public, max-age=60"
        }
    ]
},
{
    "source": "/api/wildcard-redirect",
    "headers": [
        {
            "key": "Content-Security-Policy",
            "value": "default-src 'self' https: data: blob: 'unsafe-inline' 'unsafe-eval'; frame-src *;
script-src 'self' 'unsafe-inline' 'unsafe-eval' https:; style-src 'self' 'unsafe-inline' https:;"
        },
        {
            "key": "Access-Control-Allow-Origin",
            "value": "*"
        }
    ]
}
],
"cleanUrls": false,
"trailingSlash": false
}

```

Package_Configuration

```
{
  "name": "packcdn",
  "version": "1.0.0",
  "type": "module",
  "description": "Custom package CDN system for publishing and sharing npm, Python, and
WebAssembly packages with automatic CDN generation",
  "main": "selector.html",
  "scripts": {
    "dev": "vercel dev",
    "start": "vercel dev",
    "deploy": "vercel --prod",
    "build": "echo 'No build step required for static site'",
    "test": "echo \\\"No tests specified\\\" && exit 0",
    "setup": "node scripts/setup-db.js"
  },
  "keywords": [
    "cdn",
    "package-manager",
    "npm",
    "python",
    "wasm",
    "cloudflare",
    "vercel",
    "supabase",
    "monaco-editor",
    "secure-login",
    "anonymous-auth"
  ],
  "author": "PackCDN Team",
  "license": "MIT",
  "dependencies": {
    "@supabase/supabase-js": "^2.39.0",
    "nodemailer": "^6.9.7",
    "uuid": "^9.0.1",
    "bcryptjs": "^2.4.3",
    "jsonwebtoken": "^9.0.2",
    "cookie": "^0.5.0",
    "crypto": "^1.0.1",
    "@vercel/node": "^2.15.3",
    "dotenv": "^16.3.1",
    "express-rate-limit": "^7.1.5",
    "helmet": "^7.1.0",
    "express": "^4.18.2",
    "cors": "^2.8.5"
  }
}
```

```
,
"devDependencies": {
  "vercel": "^34.0.0",
  "nodemon": "^3.0.1"
},
"engines": {
  "node": ">=18.0.0",
  "npm": ">=9.0.0"
},
"repository": {
  "type": "git",
  "url": "https://github.com/yourusername/packcdn.git"
},
"bugs": {
  "url": "https://github.com/yourusername/packcdn/issues"
},
"homepage": "https://pack-cdn.vercel.app",
"files": [
  "*.html",
  "*.js",
  "api/*.js",
  "public/*",
  "scripts/*.js"
]
}
```