CSV Grader XBlock Plugin — Complete Setup Guide

Overview

This guide documents everything needed to recreate the `csv_grader` Tutor plugin with a Studio XBlock UI from scratch on any device. Faculty can upload grades directly from Studio without ever needing a block ID or terminal access.

Environment Requirements

Requirement	Value
Tutor version	20.x (Teak)
Open edX release	Teak
Python in LMS/CMS image	3.11.8
Docker Desktop RAM	8GB minimum (10GB recommended)
Mac architecture	ARM64 (Apple Silicon)

Complete File Structure

All files live inside the Tutor plugins root. Find yours with:

```bash
tutor plugins printroot
# Example output: /Users/yourname/Library/Application Support/tutor-plugins/
```

```
<tutor-plugins-root>/
├──── csv_grader.py                    ← Plugin entry point
└──── templates/
    └──── csv_grader/                  ← Django app source
        ├──── __init__.py              ← Empty
        ├──── apps.py                  ← Django AppConfig
        ├──── xblock_csv_grader.py         ← XBlock definition
        ├──── static/
        │   ├──── css/
        │   │   └──── csv_grader.css       ← XBlock styles
        │   └──── js/
        │       └──── csv_grader.js        ← XBlock frontend logic
        ├──── templates_xblock/
        │   └──── csv_grader.html          ← XBlock Studio UI template
        └──── management/
            ├──── __init__.py          ← Empty
            └──── commands/
                ├──── __init__.py      ← Empty
                └──── import_csv_grades.py     ← CLI management command
```

Step 1 — Create Folder Structure

```bash
cd "$(tutor plugins printroot)"
mkdir -p templates/csv_grader/static/css
mkdir -p templates/csv_grader/static/js
mkdir -p templates/csv_grader/templates_xblock
mkdir -p templates/csv_grader/management/commands
touch templates/csv_grader/__init__.py
touch templates/csv_grader/management/__init__.py
touch templates/csv_grader/management/commands/__init__.py
```

Step 2 — Create `csv_grader.py`

Path: `<tutor-plugins-root>/csv_grader.py`

```python
```

```
import os
from tutor import hooks

# Register templates folder
template_folder = os.path.join(os.path.dirname(__file__), "templates")
hooks.Filters.ENV_TEMPLATE_ROOTS.add_item(template_folder)

# Render templates/csv_grader/ into env/build/openedx/csv_grader_app/
hooks.Filters.ENV_TEMPLATE_TARGETS.add_item(
    ("csv_grader", "build/openedx/csv_grader_app")
)

# Add to LMS INSTALLED_APPS
hooks.Filters.ENV_PATCHES.add_item((
    "openedx-lms-common-settings",
    'INSTALLED_APPS += ["csv_grader"]'
))

# Add to CMS INSTALLED_APPS and register XBlock in Advanced Component menu
hooks.Filters.ENV_PATCHES.add_item((
    "openedx-cms-common-settings",
    """INSTALLED_APPS += ["csv_grader"]
ADVANCED_COMPONENT_TYPES = list(globals().get("ADVANCED_COMPONENT_TYPES", []))
if "csv_grader" not in ADVANCED_COMPONENT_TYPES:
    ADVANCED_COMPONENT_TYPES += ["csv_grader"]"""
))

# Copy app into image and register XBlock entry point via pip
hooks.Filters.ENV_PATCHES.add_item((
    "openedx-dockerfile-post-python-requirements",
    r"""COPY --chown=app:app ./csv_grader_app/csv_grader /openedx/venv/lib/python3.11/site-packages/csv_grader
RUN mkdir -p /tmp/csvpkg && \
    printf '[metadata]\nname = csv-grader\nversion = 0.1\n\n[options]\npackages = find:\npackage_dir = =src\n\n[options.entry
    printf 'from setuptools import setup\nsetup()\n' > /tmp/csvpkg/setup.py && \
    mkdir -p /tmp/csvpkg/src && \
    ln -s /openedx/venv/lib/python3.11/site-packages/csv_grader /tmp/csvpkg/src/csv_grader && \
    cd /tmp/csvpkg && /openedx/venv/bin/pip install -e . --no-deps -q"""
))
```

Step 3 — Create `apps.py`

Path: `templates/csv_grader/apps.py`

```
python
```

```python
from django.apps import AppConfig


class CsvGraderConfig(AppConfig):
    name = "csv_grader"
    verbose_name = "CSV Grader"
    default_auto_field = "django.db.models.BigAutoField"
```

Step 4 — Create `xblock_csv_grader.py`

Path: `templates/csv_grader/xblock_csv_grader.py`

> **Critical:** Do NOT put `get_user_model()` at module level — it must be inside the method or Django crashes on import.

python

```python
import csv
import io
import json
import logging

from web_fragments.fragment import Fragment
from xblock.core import XBlock
from xblock.fields import Scope, String, Integer
from xblockutils.resources import ResourceLoader

log = logging.getLogger(__name__)
loader = ResourceLoader(__name__)


@XBlock.needs("i18n")
class CsvGraderXBlock(XBlock):

    display_name = String(
        display_name="Display Name",
        default="CSV Grade Importer",
        scope=Scope.settings,
    )

    last_import_summary = String(default="", scope=Scope.content)
    last_import_count = Integer(default=0, scope=Scope.content)

    def resource_string(self, path):
        return loader.load_unicode(path)

    def student_view(self, context=None):
        return Fragment("<div style='display:none'>CSV Grader (Studio only)</div>")

    def studio_view(self, context=None):
        problem_blocks = self._get_course_problems()

        options_html = ""
        for block in problem_blocks:
            options_html += "<option value='{bid}'>{name}</option>".format(
                bid=str(block["usage_key"]),
                name=block["display_name"]
            )

        html = self.resource_string("templates_xblock/csv_grader.html")
        frag = Fragment(html.format(
            block_id=str(self.location),
            options=options_html,
            last_summary=self.last_import_summary or "",
        ))
        frag.add_css(self.resource_string("static/css/csv_grader.css"))
        frag.add_javascript(self.resource_string("static/js/csv_grader.js"))
        frag.initialize_js("CsvGraderXBlock")
        return frag

    def _get_course_problems(self):
        try:
            from xmodule.modulestore.django import modulestore
            course_key = self.location.course_key
            blocks = modulestore().get_items(
                course_key,
                qualifiers={"category": "problem"}
            )
            result = []
            for block in blocks:
                result.append({
                    "usage_key": block.location,
                    "display_name": block.display_name or str(block.location),
                })
            return result
        except Exception as e:
            log.error("csv_grader: could not fetch problem blocks: %s", e)
            return []

    @XBlock.json_handler
    def import_grades(self, data, suffix=""):
        # All imports inside method — Django must be ready first
        from django.contrib.auth import get_user_model
        from lms.djangoapps.courseware.models import StudentModule
        from opaque_keys.edx.keys import UsageKey

        User = get_user_model()

        csv_content = data.get("csv_content", "")
        target_block = data.get("target_block", "").strip()

        if not csv_content:
            return {"success": False, "error": "No CSV data received"}
        if not target_block:
            return {"success": False, "error": "Please select a target problem block"}

        try:
            usage_key = UsageKey.from_string(target_block)
        except Exception:
            return {"success": False, "error": "Invalid block ID: " + target_block}
    course_key = usage_key.course_key
    max_grade = float(data.get("max_grade", 1.0))

    results = []
    errors = []
    created_count = 0
    updated_count = 0

    reader = csv.reader(io.StringIO(csv_content))
    for line_num, row in enumerate(reader, 1):
        if not row or len(row) < 2:
            continue
        username = row[0].strip()
        grade_str = row[1].strip()

        try:
            user = User.objects.get(username=username)
            grade = float(grade_str)
        except Exception as e:
            errors.append("Line {}: {}".format(line_num, str(e)))
            continue

        obj, created = StudentModule.objects.update_or_create(
            student=user,
            course_id=course_key,
            module_state_key=usage_key,
            defaults={
                "grade": grade,
                "max_grade": max_grade,
                "module_type": "problem",
                "state": json.dumps({
                    "score": {
                        "raw_earned": grade,
                        "raw_possible": max_grade
                    }
                }),
            }
        )
        if created:
            created_count += 1
        else:
            updated_count += 1
        results.append({
            "username": username,
            "grade": grade,
            "action": "created" if created else "updated"
            })

        summary = "{} created, {} updated".format(created_count, updated_count)
        if errors:
            summary += ", {} errors".format(len(errors))

        self.last_import_summary = summary
        self.last_import_count = len(results)

        return {
            "success": True,
            "summary": summary,
            "results": results,
            "errors": errors,
            "created": created_count,
            "updated": updated_count,
        }

    @staticmethod
    def workbench_scenarios():
        return [("CsvGraderXBlock", "<csv-grader/>")]
```

Step 5 — Create `templates_xblock/csv_grader.html`

Path: `templates/csv_grader/templates_xblock/csv_grader.html`

```
html
```

```html
<div class="csvgrader-wrap" id="csvgrader-wrap">
 <div class="csvgrader-header">
  <span class="csvgrader-icon">📊 </span>
  <div>
   <div class="csvgrader-title">CSV Grade Importer</div>
   <div class="csvgrader-block-id">{block_id}</div>
  </div>
  <div class="csvgrader-badge">Studio Only</div>
 </div>
 <div class="csvgrader-body">
  <div class="csvgrader-section">
   <label class="csvgrader-label">Target Problem Block</label>
   <select id="csvgrader-target" class="csvgrader-select">
    <option value="">-- Select a problem block --</option>
    {options}
   </select>
   <div class="csvgrader-hint">Grades will be written into the selected block</div>
  </div>
  <div class="csvgrader-section">
   <label class="csvgrader-label">Max Grade</label>
   <input type="number" id="csvgrader-maxgrade" value="1" min="0.1" step="0.1" class="csvgrader-input" style="width:1
  </div>
  <div class="csvgrader-section">
   <label class="csvgrader-label">Upload CSV File</label>
   <div class="csvgrader-dropzone" id="csvgrader-dropzone">
    <input type="file" id="csvgrader-file" accept=".csv,.txt" />
    <div class="csvgrader-drop-icon">📂 </div>
    <div class="csvgrader-drop-text">Drop CSV here or click to browse</div>
    <div class="csvgrader-drop-sub">Format: username,grade — one per line, no header</div>
   </div>
  </div>
  <div class="csvgrader-preview" id="csvgrader-preview" style="display:none">
   <div class="csvgrader-stats" id="csvgrader-stats"></div>
   <div class="csvgrader-table-wrap">
    <table class="csvgrader-table">
     <thead><tr><th>#</th><th>Username</th><th>Grade</th></tr></thead>
     <tbody id="csvgrader-tbody"></tbody>
    </table>
   </div>
  </div>
  <div class="csvgrader-actions">
   <button id="csvgrader-btn" class="csvgrader-btn" disabled>▶ Import Grades</button>
   <span id="csvgrader-spinner" class="csvgrader-spinner" style="display:none">⏳ Importing...</span>
  </div>
  <div id="csvgrader-result" class="csvgrader-result" style="display:none"></div>
  <div class="csvgrader-last" id="csvgrader-last">{last_summary}</div>
  </div>
</div>
```

Step 6 — Create `static/css/csv_grader.css`

Path: `templates/csv_grader/static/css/csv_grader.css`

```
css
```

```css
.csvgrader-wrap {
  font-family: 'Inter', sans-serif;
  background: #1a1a2e;
  border: 1px solid #2a2a4a;
  border-radius: 12px;
  overflow: hidden;
  max-width: 600px;
  margin: 16px auto;
  color: #e0e0f0;
}
.csvgrader-header {
  display: flex;
  align-items: center;
  gap: 12px;
  background: #12122a;
  padding: 16px 20px;
  border-bottom: 1px solid #2a2a4a;
}
.csvgrader-icon { font-size: 24px; }
.csvgrader-title { font-size: 15px; font-weight: 700; }
.csvgrader-block-id {
  font-size: 10px;
  font-family: monospace;
  color: #6666aa;
  margin-top: 2px;
  word-break: break-all;
}
.csvgrader-badge {
  margin-left: auto;
  background: rgba(124,92,252,0.2);
  color: #a080ff;
  border: 1px solid rgba(124,92,252,0.3);
  border-radius: 20px;
  padding: 3px 10px;
  font-size: 11px;
  white-space: nowrap;
}
.csvgrader-body { padding: 20px; }
.csvgrader-section { margin-bottom: 18px; }
.csvgrader-label {
  display: block;
  font-size: 11px;
  text-transform: uppercase;
  letter-spacing: 0.5px;
  color: #6666aa;
  margin-bottom: 8px;
  font-family: monospace;
}
.csvgrader-input {
  background: #12122a;
  border: 1px solid #2a2a4a;
  border-radius: 6px;
  padding: 7px 12px;
  color: #e0e0f0;
  font-size: 14px;
  outline: none;
}
.csvgrader-input:focus { border-color: #7c5cfc; }
.csvgrader-select {
  width: 100%;
  background: #12122a;
  border: 1px solid #2a2a4a;
  border-radius: 6px;
  padding: 9px 12px;
  color: #e0e0f0;
  font-size: 13px;
  font-family: monospace;
  outline: none;
  cursor: pointer;
}
.csvgrader-select:focus { border-color: #7c5cfc; }
.csvgrader-hint { font-size: 11px; color: #6666aa; margin-top: 5px; font-family: monospace; }
.csvgrader-dropzone {
  border: 2px dashed #2a2a4a;
  border-radius: 10px;
  padding: 28px;
  text-align: center;
  cursor: pointer;
  position: relative;
  background: #12122a;
  transition: all 0.2s;
}
.csvgrader-dropzone:hover, .csvgrader-dropzone.drag-over {
  border-color: #7c5cfc;
  background: rgba(124,92,252,0.05);
}
.csvgrader-dropzone input[type="file"] {
  position: absolute; inset: 0; opacity: 0;
  cursor: pointer; width: 100%; height: 100%;
}
.csvgrader-drop-icon { font-size: 28px; margin-bottom: 8px; }
.csvgrader-drop-text { font-size: 13px; font-weight: 600; }
.csvgrader-drop-sub { font-size: 11px; color: #6666aa; margin-top: 4px; font-family: monospace; }
```

```
.csvgrader-stats { display: flex; gap: 10px; margin-bottom: 12px; }
.csvgrader-stat {
  flex: 1;
  background: #12122a;
  border: 1px solid #2a2a4a;
  border-radius: 8px;
  padding: 10px;
  text-align: center;
}
.csvgrader-stat-num { font-size: 22px; font-weight: 800; font-family: monospace; }
.csvgrader-stat-label { font-size: 10px; color: #6666aa; text-transform: uppercase; margin-top: 3px; }
.csvgrader-table-wrap {
  border: 1px solid #2a2a4a;
  border-radius: 8px;
  overflow: hidden;
  max-height: 180px;
  overflow-y: auto;
  margin-bottom: 16px;
}
.csvgrader-table { width: 100%; border-collapse: collapse; font-size: 12px; font-family: monospace; }
.csvgrader-table th {
  padding: 7px 12px;
  text-align: left;
  color: #6666aa;
  background: #12122a;
  font-size: 10px;
  text-transform: uppercase;
  letter-spacing: 0.5px;
}
.csvgrader-table td { padding: 7px 12px; border-top: 1px solid #1e1e3a; }
.csvgrader-actions { display: flex; align-items: center; gap: 12px; }
.csvgrader-btn {
  background: linear-gradient(135deg, #7c5cfc, #9b7cff);
  color: white;
  border: none;
  border-radius: 8px;
  padding: 10px 22px;
  font-size: 14px;
  font-weight: 700;
  cursor: pointer;
  transition: all 0.2s;
  box-shadow: 0 4px 14px rgba(124,92,252,0.35);
}
.csvgrader-btn:hover { transform: translateY(-1px); box-shadow: 0 6px 18px rgba(124,92,252,0.5); }
.csvgrader-btn:disabled { opacity: 0.4; cursor: not-allowed; transform: none; box-shadow: none; }
.csvgrader-spinner { font-size: 13px; color: #a080ff; }
.csvgrader-result {
  margin-top: 14px;
  padding: 14px 16px;
  border-radius: 8px;
  font-size: 13px;
  font-family: monospace;
  line-height: 1.7;
}
.csvgrader-result.success {
  background: rgba(92,252,160,0.08);
  border: 1px solid rgba(92,252,160,0.25);
  color: #5cfca0;
}
.csvgrader-result.error {
  background: rgba(252,92,125,0.08);
  border: 1px solid rgba(252,92,125,0.25);
  color: #fc5c7d;
}
.csvgrader-last { margin-top: 12px; font-size: 12px; font-family: monospace; color: #6666aa; }
```

Step 7 — Create `static/js/csv_grader.js`

Path: `templates/csv_grader/static/js/csv_grader.js`

```javascript
```

```
function CsvGraderXBlock(runtime, element) {
  var parsedRows = [];
  var csvContent = "";
  var $el = $(element);
  function find(sel) { return $el.find(sel); }

  find("#csvgrader-file").on("change", function() {
    if (this.files && this.files[0]) processFile(this.files[0]);
  });

  find("#csvgrader-dropzone").on("dragover", function(e) {
    e.preventDefault(); $(this).addClass("drag-over");
  }).on("dragleave", function() {
    $(this).removeClass("drag-over");
  }).on("drop", function(e) {
    e.preventDefault(); $(this).removeClass("drag-over");
    var f = e.originalEvent.dataTransfer.files[0];
    if (f) processFile(f);
  });

  function processFile(file) {
    var reader = new FileReader();
    reader.onload = function(e) {
      csvContent = e.target.result;
      parsedRows = [];
      csvContent.split("\n").filter(function(l){ return l.trim(); }).forEach(function(line) {
        var parts = line.split(",");
        if (parts.length >= 2)
          parsedRows.push({ username: parts[0].trim(), grade: parseFloat(parts[1].trim()) || 0 });
      });
      showPreview();
      updateBtn();
    };
    reader.readAsText(file);
  }

  function showPreview() {
    var pass = parsedRows.filter(function(r){ return r.grade > 0; }).length;
    find("#csvgrader-stats").html(
      "<div class='csvgrader-stat'><div class='csvgrader-stat-num' style='color:#a080ff'>" + parsedRows.length + "</div><div c
      "<div class='csvgrader-stat'><div class='csvgrader-stat-num' style='color:#5cfca0'>" + pass + "</div><div class='csvgrade
      "<div class='csvgrader-stat'><div class='csvgrader-stat-num' style='color:#fc5c7d'>" + (parsedRows.length - pass) + "</di
    );
    find("#csvgrader-tbody").html(parsedRows.map(function(r, i) {
      return "<tr><td>" + (i+1) + "</td><td>" + r.username + "</td><td>" + r.grade + "</td></tr>";
    }).join(""));
    find("#csvgrader-preview").show();
  }

  function updateBtn() {
    var hasFile = parsedRows.length > 0;
    var hasTarget = find("#csvgrader-target").val() !== "";
    find("#csvgrader-btn").prop("disabled", !(hasFile && hasTarget));
  }

  find("#csvgrader-target").on("change", updateBtn);

  find("#csvgrader-btn").on("click", function() {
    var target = find("#csvgrader-target").val();
    if (!csvContent || parsedRows.length === 0) { alert("Please select a CSV file first."); return; }
    if (!target) { alert("Please select a target problem block."); return; }

    find("#csvgrader-btn").prop("disabled", true);
    find("#csvgrader-spinner").show();
    find("#csvgrader-result").hide();

    $.ajax({
      type: "POST",
      url: runtime.handlerUrl(element, "import_grades"),
      data: JSON.stringify({
        csv_content: csvContent,
        max_grade: parseFloat(find("#csvgrader-maxgrade").val()) || 1.0,
        target_block: target
      }),
      contentType: "application/json",
      success: function(response) {
        find("#csvgrader-spinner").hide();
        find("#csvgrader-btn").prop("disabled", false);
        if (response.success) {
          var html = "<strong>✓ " + response.summary + "</strong><br><br>";
          response.results.forEach(function(r) {
            html += (r.action === "created" ? "+ " : "↻ ") + r.username + ": " + r.grade + "<br>";
          });
          if (response.errors && response.errors.length) {
            html += "<br><span style='color:#fc5c7d'>Errors:</span><br>";
            response.errors.forEach(function(e) { html += "✗ " + e + "<br>"; });
          }
          find("#csvgrader-result").removeClass("error").addClass("success").html(html).show();
          find("#csvgrader-last").text("Last import: " + response.summary);
        } else {
          find("#csvgrader-result").removeClass("success").addClass("error").html("✗ " + (response.error || "Unknown error")).sh
        }
      },
      error: function(xhr) {
        find("#csvgrader-spinner").hide();
        find("#csvgrader-btn").prop("disabled", false);
        find("#csvgrader-result").removeClass("success").addClass("error").html("✗ Request failed (" + xhr.status + ")").show();
      }
    });
   });
  }
```

Step 8 — Create `management/commands/import_csv_grades.py`

Path: `templates/csv_grader/management/commands/import_csv_grades.py`

> **Important:** No `{% raw %}` tags — this file is copied directly, not rendered by Jinja2.

```
python
```

```python
import csv
from django.core.management.base import BaseCommand
from django.contrib.auth import get_user_model
from opaque_keys.edx.keys import CourseKey, UsageKey
from lms.djangoapps.courseware.models import StudentModule

User = get_user_model()

class Command(BaseCommand):
    help = "Import grades from a CSV into a problem block via StudentModule"

    def add_arguments(self, parser):
        parser.add_argument("--csv", required=True, help="Path to marks CSV file")
        parser.add_argument("--block", required=True, help="Usage key of the problem block")
        parser.add_argument("--course", required=True, help="Course key e.g. course-v1:DEMO+CSV101+2026")
        parser.add_argument("--max-grade", type=float, default=1.0)

    def handle(self, *args, **options):
        course_key = CourseKey.from_string(options["course"])
        usage_key  = UsageKey.from_string(options["block"])
        max_grade  = options["max_grade"]

        with open(options["csv"], newline="") as f:
            for row in csv.reader(f):
                if len(row) < 2:
                    continue
                username, grade_str = row[0].strip(), row[1].strip()
                try:
                    user  = User.objects.get(username=username)
                    grade = float(grade_str)
                except (User.DoesNotExist, ValueError) as e:
                    self.stderr.write("Skipping {}: {}".format(username, e))
                    continue

                obj, created = StudentModule.objects.update_or_create(
                    student=user,
                    course_id=course_key,
                    module_state_key=usage_key,
                    defaults={
                        "grade": grade,
                        "max_grade": max_grade,
                        "module_type": "problem",
                        "state": '{"score": {"raw_earned": ' + str(grade) + ', "raw_possible": ' + str(max_grade) + '}}',
                    }
                )
                verb = "Created" if created else "Updated"
            self.stdout.write("{} grade for {}: {}/{}".format(verb, username, grade, max_grade))

        self.stdout.write(self.style.SUCCESS("Done!"))
```

Step 9 — Enable Plugin and Render Environment

```bash
tutor plugins enable csv_grader
tutor plugins list   # confirm csv_grader shows as enabled

tutor config save

# Verify files rendered into build context
find "$(tutor config printroot)/env/build/openedx/csv_grader_app" -type f

# Verify COPY line is in Dockerfile
grep "csv_grader" "$(tutor config printroot)/env/build/openedx/Dockerfile"
```

Expected Dockerfile output:

```
COPY --chown=app:app ./csv_grader_app/csv_grader /openedx/venv/lib/python3.11/site-packages/csv_grader
RUN mkdir -p /tmp/csvpkg && ...
```

Step 10 — Build the Docker Image

> **Before building:** Set Docker Desktop memory to 8GB+ Docker Desktop → Settings → Resources → Memory → 8GB → Apply & Restart

```bash
export NODE_OPTIONS="--max-old-space-size=4096"
tutor images build openedx
```

This takes 15-30 minutes. Most steps are cached on repeat builds.

When to rebuild the image:

- First time setting up the plugin on a new machine
- After any changes to plugin source files
- After adding new Python dependencies
- Never needed just for content/course changes

Step 11 — Start Platform

```bash
tutor local start -d
# Wait 30 seconds for all containers to be healthy
docker ps --filter name=tutor_local --format "{{.Names}} {{.Status}}"
```

All containers should show `Up X seconds` not `Restarting`.

Step 12 — Verify Installation

```bash
# Check app is in LMS
tutor local run lms ls /openedx/venv/lib/python3.11/site-packages/csv_grader/

# Check XBlock entry point is registered
docker exec tutor_local-cms-1 python -c "
import pkg_resources
for ep in pkg_resources.iter_entry_points('xblock.v1'):
    if 'csv' in ep.name:
        print('FOUND:', ep)
"

# Check ADVANCED_COMPONENT_TYPES includes csv_grader
tutor local run cms ./manage.py cms shell -c "
from django.conf import settings
types = getattr(settings, 'ADVANCED_COMPONENT_TYPES', [])
print('csv_grader registered:', 'csv_grader' in types)
"
```

Step 13 — Configure Course in Studio

1. Go to `http://studio.local.openedx.io`
2. Open your course
3. Go to **Settings → Advanced Settings**
4. Find **Advanced Module List**
5. Set it to: `["csv_grader"]`
6. Click **Save Changes**

> This must be done once per course. It whitelists the XBlock for that specific course.

Step 14 — Use the XBlock

1. Open any Unit in Studio
2. Click **Add Component → Advanced → CSV Grade Importer**
3. Click the **pencil/edit icon** on the new block
4. Select the **Target Problem Block** from the dropdown
5. Set **Max Grade**
6. Drop your CSV file (format: `username,grade` per line, no header)
7. Preview appears showing all students
8. Click **▶ Import Grades**
9. Results show inline with created/updated counts

Step 15 — Verify Grades Were Written

bash

```bash
docker exec $(docker ps --filter name=tutor_local-lms-1 --format '{{.ID}}') ./manage.py lms shell -c "
from lms.djangoapps.courseware.models import StudentModule
from opaque_keys.edx.keys import UsageKey
usage_key = UsageKey.from_string('YOUR_BLOCK_ID_HERE')
records = StudentModule.objects.filter(module_state_key=usage_key)
print('username, grade, max_grade')
for r in records:
    print(r.student.username, r.grade, r.max_grade)
"
```

After Every Docker Restart (Until Permanent Build)

If you haven't done a full image rebuild yet, run these after every `tutor local start`:

bash

```
# Re-copy app into containers
docker cp "$(tutor config printroot)/env/build/openedx/csv_grader_app/csv_grader" \
  "tutor_local-cms-1:/openedx/venv/lib/python3.11/site-packages/csv_grader"

docker cp "$(tutor config printroot)/env/build/openedx/csv_grader_app/csv_grader" \
  "tutor_local-lms-1:/openedx/venv/lib/python3.11/site-packages/csv_grader"

# Re-register XBlock entry point
docker exec tutor_local-cms-1 bash -c 'rm -rf /tmp/csvpkg && mkdir -p /tmp/csvpkg/src && ln -sf /openedx/venv/lib/python3
```

Recalculate Grades in Gradebook

After importing, trigger grade recalculation so the LMS gradebook reflects the new scores:

```bash
docker exec $(docker ps --filter name=tutor_local-lms-1 --format '{{.ID}}') ./manage.py lms shell -c "
from lms.djangoapps.grades.tasks import recalculate_course_and_subsection_grades_for_course
from opaque_keys.edx.keys import CourseKey
course_key = CourseKey.from_string('course-v1:DEMO+CSV101+2026')
recalculate_course_and_subsection_grades_for_course.delay(str(course_key))
print('Grade recalculation queued')
"
```

Troubleshooting

CMS crash: `ModuleNotFoundError: No module named 'csv_grader'`

The app wasn't copied into the image. Run the "After Every Docker Restart" commands above.

XBlock not in Advanced menu

Either the entry point isn't registered (run the re-register command) or the course Advanced Module List doesn't include `csv_grader` (check Studio → Settings → Advanced Settings).

Edit dialog shows blank / `xblockElement is empty`

uWSGI has cached old bytecode. Clear it:

```bash
docker exec -u root tutor_local-cms-1 bash -c 'find /openedx/venv/lib/python3.11/site-packages/csv_grader -name "*.pyc" -de
docker stop tutor_local-cms-1 && docker start tutor_local-cms-1
```

Then re-copy files and re-register entry point.

`get_user_model()` crash on import

The `xblock_csv_grader.py` has `User = get_user_model()` at module level. It must be inside the `import_grades` method. See Step 4.

webpack `cannot allocate memory` during build

```bash
export NODE_OPTIONS="--max-old-space-size=4096"
```

Set Docker Desktop RAM to 8-10GB and rebuild.

Grades written but gradebook shows 0

Run the grade recalculation command in the section above.

Key Paths Reference

Item	Path
Plugin entry point	`$(tutor plugins printroot)/csv_grader.py`
All plugin source files	`$(tutor plugins printroot)/templates/csv_grader/`
Rendered build context	`$(tutor config printroot)/env/build/openedx/csv_grader_app/`
Dockerfile	`$(tutor config printroot)/env/build/openedx/Dockerfile`
LMS settings	`$(tutor config printroot)/env/apps/openedx/settings/lms/production.py`
CMS settings	`$(tutor config printroot)/env/apps/openedx/settings/cms/production.py`
App inside containers	`/openedx/venv/lib/python3.11/site-packages/csv_grader/`