ChatGPT

1. OCR + AI Integration

- **Issue: Limited OCR usage.** The backend's OCR (`vision.ImageAnnotatorClient().text_detection`) only returns raw text [1]. It doesn't use specialized modes (e.g. `document_text_detection` for PDFs/multi-page docs), so layout or complex text may be lost.
  Expected: All text from survey plans/deeds (including multi-page PDFs) should be captured. Use Google Vision's full document text detection.
  Fix: Switch to Vision's `document_text_detection` for PDFs and complex layouts. For example:

```
# In extract_text (backend/app/services/ocr_processing.py)
if file_model.mimetype == 'application/pdf':
    response = vision_client.document_text_detection(file_content)
else:
    response = vision_client.text_detection(file_content)
texts = response.text_annotations
text = texts[0].description if texts else ""
```

  This ensures multi-page PDF OCR. *(Uses Google Vision API key) {Dependencies: Google Cloud Vision [1] }*
  Priority: High.

- **Issue: Incomplete AI prompts / missing fields.** The GPT-4 prompts cover many fields, but some expected data aren't extracted. For example, the **Certificate of Sale** (a deed-like document) isn't specifically handled – any such doc is classified as `DocumentType.OTHER` and only gets a generic note [2]. Also, prompts for *road access* and *distances* appear only in the comprehensive extractor, not the survey/deed-specific extractors.
  Expected: Key info from survey plans and deeds – Lot No, Plan No/Date, Surveyor, Boundaries, Coordinates, GN/DS/Province, Land Area, Road Access, Ownership – should all be extracted. Certificate of Sale or similar legal docs should populate deed-like fields.
  Fix: Enhance the AI extraction service. For example:

- Add a new case for Certificate-of-Sale documents in `detect_document_type` or augment the Deed prompt.
- Ensure GPT prompts explicitly ask for "road_access" and "distance_to_nearest_city/town". (The `extract_comprehensive_property_data` prompt already includes these [3].)

- Example code:

```
# In ai_extraction.py, extend deed extraction prompt
prompt = f"""
You are an expert at extracting information from Sri Lankan property deeds
```

```
and sale certificates.
...
Extract and return JSON with these fields (use null if missing):
  "deed_number", "deed_date", "notary_attorney", "vendor", "purchaser",
  "lot_number", "plan_number", "road_access", "land_area",
"encumbrances", ...
"""
```

And handle it in `extract_deed_data`.
(Dependencies: OpenAI GPT-4 model, Google Vision OCR [4] [3])
Priority: High.

- **Issue: Batch processing limit / missing multi-file support.** The `/batch-process` endpoint forbids more than 10 files [5] , which may surprise users. Also, the frontend has no flow to upload multiple docs at once or trigger batch parsing.
Expected: Users should be able to upload multiple related docs (survey plan, deed, etc.) in one batch for cross-validation.
Fix: Adjust or remove the 10-file cap (or surface it to users), and build a UI flow. For example, in the appendices/upload step, call the batch endpoint:

```
// In frontend (after user selects multiple file IDs):
const res = await reportsAPI.batchProcessDocuments({ file_ids:
selectedFileIds, consolidate_analysis: true, auto_populate: true });
if (res.auto_population_data) {
  populateFromAiAnalysis({ document_analysis: res }); // merge AI results
into wizard
}
```

This uses the wizard's `populateFromAiAnalysis` to push extracted fields into the form [6] [7] .
(The code above expects a similar shape.)
Dependencies: Google Vision, OpenAI, FastAPI backend.
Priority: Medium.

- **Issue: AI results not populating wizard fields.** The frontend isn't automatically merging OCR/GPT fields into the form. While the `WizardProvider` has a `populateFromAiAnalysis()` method that uses a smart merger (and falls back to per-step updates) [6] [7] , it must be explicitly invoked with the AI output.
Expected: Once OCR+AI runs, relevant form fields auto-fill. E.g. Lot No→Identification, boundaries→Site, directions/distance→Location, etc.
Fix: After batch OCR, call the wizard context's `populateFromAiAnalysis`. For example:

```
import { useWizard } from '@/components/wizard/WizardProvider';
const { populateFromAiAnalysis } = useWizard();
// ...
```

```
const result = await mapsAPI.batchProcess(...);
populateFromAiAnalysis(result);  // Merges into steps
```

This triggers the code at [73–74], which uses `updateStepData` under the hood [8] [7]. If using the comprehensive extractor, ensure its output is forwarded (the fallback looks for `document_analysis.comprehensive_data` [9]).
(Dependencies: Wizard context in frontend) **Priority:** High.

2. Location Intelligence

- **Issue: Reverse geocoding missing fine divisions.** The `reverse_geocode` function only returns province, district, city/area [10] – it has no Sri Lankan GN or DS division fields. GN/DS data are expected (especially in reports) but Google's API doesn't supply them.
Expected: Administrative divisions (GN, DS, Province) should populate Location fields.
Fix: Integrate a Sri Lanka-specific geodata source. Options: call a GIS service or include a local geodatabase (e.g. shapefiles of GN/DS boundaries) to map lat/lng. For example: after reverse-geocoding, lookup the lat/lng in a DS/GN dataset and set `location.gn_division`, `location.ds_division`. (This requires a dependency like a GIS library or custom REST endpoint.)
E.g.:

```
# Pseudocode in backend or API:
ds = lookup_ds_division(latitude, longitude)
gn = lookup_gn_division(latitude, longitude)
return {
    "formatted_address": ...,
    "components": {...},
    "ds_division": ds,
    "gn_division": gn
}
```

Then in frontend's `reverseGeocodeLocation`, include these in `location` update.
Dependencies: Possibly a GIS database or Google Maps' "sublocality_level_2" if it maps.
Priority: Medium.

- **Issue: Nearby POIs not fetched/displayed.** The system has `find_nearby_places` and `find_nearby_amenities` for schools, hospitals, etc. (see [25†L430-L438]) but the UI does not trigger these. For instance, the Locality step shows nearest school/bank, but no code calls `mapsAPI.getNearbyAmenities`.
Expected: When requested, the app should query Google Places for e.g. schools, banks, supermarkets within ~5km, and populate list/fields (like nearest school).
Fix: Add frontend calls to the Places endpoints. For example, in **LocalityStep.tsx** or **LocationStep.tsx**:

```
const amenities = await mapsAPI.getNearbyAmenities(lat, lng);
updateStepData('locality', {
  nearest_school: amenities.amenities.schools.places[0]?.name,
  nearest_hospital: amenities.amenities.hospitals.places[0]?.name,
  // ... etc.
});
```

This uses the backend's amenity search (which returns up to 5 closest for each category) [11].
Dependencies: Google Places API key (already in settings) [12].
Priority: Low.

- **Issue: Static map not updating / satellite view blank.** The static map URL is generated via `generate_static_map_url`, which defaults to `maptype=roadmap` [13]. The frontend allows changing `mapType` (roadmap vs satellite) and re-calls the API, but if nothing was called initially the map may be blank. Some users report no satellite image.
Expected: Static map image should display in Location step, both road and satellite views.
Fix: Ensure the static-map endpoint is invoked once coordinates are set. For example, after reverse-geocoding or coordinate input:

```
const mapData = await mapsAPI.generateStaticMap(lat, lng, { zoom:15,
mapType:'roadmap' });
setStaticMapUrl(mapData.map_url);
```

Then allow toggle to `'satellite'`. The backend handles it (see [23†L47-L53] and [24†L247-L254]). If satellite still returns blank, check API key restrictions.
Dependencies: Google Static Maps API [12].
Priority: Medium.

- **Issue: Road access details incomplete.** While `generate_route_description` produces a text and distance from the city, the system doesn't explicitly record road names/types in form fields. The Site/Location steps expect "access road" or "road quality" fields.
Expected: Key road names or descriptions (e.g. "Frontage on A-road, tarred") should fill SiteStep.
Fix: Extract first step of directions (from Google) for main road names. E.g. in `generate_route_description`, parse `steps[0].html_instructions` (removing HTML) and write to `location.access_road`. For instance:

```
// After getting steps in generate_route_description:
const mainRoad = leg.steps[0].html_instructions.replace(/<[^>]+>/g, '');
return { ..., "access_road": mainRoad, ... };
```

In frontend, capture that with `populateFromAiAnalysis` (it already tries `access_details.road_type` [14]). If missing, manually do in code.

Dependencies: Google Directions API [15].
Priority: Low.

3. Workflow Completion (Data Propagation)

- **Issue: Extracted data not flowing into steps.** As noted, AI fields exist but weren't being applied. The `WizardProvider` logic (in `populateFromAiAnalysis`) shows exactly how various fields map: Identification (lot, plan, boundaries, owner, deed) [6] [16]; Location (address, GN/DS, coords) [7]; Site (topography, access road) [17]; Buildings (type, area, year) [18]; Utilities and Locality (amenities, landmarks) [19]; Legal (ownership, encumbrances) [20]. If any of these are missing, the merge must have failed.
  Expected: A clean propagation so the user sees AI-suggested values in each wizard step.
  Fix: Ensure the front-end invokes `updateStepData` or `populateFromAiAnalysis` as shown above after OCR/AI. If needed, manually set fields. For example, to set boundaries in SiteStep:

  ```
  updateStepData('site', { boundaries: aiData.extracted_data.boundaries });
  ```

  Or simply rely on `populateFromAiAnalysis(aiResult.document_analysis)` so the above mapping runs. The mapping code in [73–74] can be adjusted if new fields were added.
  Dependencies: None beyond existing AI output and context.
  Priority: High.

External Dependencies Summary: The fixes rely on Google Cloud Vision API (OCR), OpenAI GPT-4 (AI parsing), Google Maps APIs (Static Maps, Geocoding, Places, Directions), and the frontend React context (WizardProvider). Citations above show how these are used in the code [1] [4] [12] [8], guiding the necessary changes.

---

[1] [5] batch_ocr.py
https://github.com/Malith-nethsiri/valuerpro-project/blob/55f7e60cdb2f05d8606d378e66193fecabec4e5a/backend/app/api/api_v1/endpoints/batch_ocr.py

[2] [3] [4] ai_extraction.py
https://github.com/Malith-nethsiri/valuerpro-project/blob/55f7e60cdb2f05d8606d378e66193fecabec4e5a/backend/app/services/ai_extraction.py

[6] [7] [8] [9] [14] [16] [17] [18] [19] [20] WizardProvider.tsx
https://github.com/Malith-nethsiri/valuerpro-project/blob/55f7e60cdb2f05d8606d378e66193fecabec4e5a/frontend/src/components/wizard/WizardProvider.tsx

[10] [11] [12] [13] [15] google_maps.py
https://github.com/Malith-nethsiri/valuerpro-project/blob/55f7e60cdb2f05d8606d378e66193fecabec4e5a/backend/app/services/google_maps.py