MEMORANDUM

DATE:	10/15/2025
FROM:	Gregg Wolin
TO:	Brett Duarte, Sang Pil Han (W.P. Carey School of Business)
RE:	"Landscape' platform

Thanks for taking the time to zoom last week and I apologize for the tardy delivery of this memo. I really appreciated your thoughts and willingness to look at the platform I am working on and how parts or all if it might become a capstone project for a group of students. While we discussed some of it last week, this memo should give you something to ponder before we reconvene to discuss next steps.

1. The Excel > Argus Workflow Gap. Every real estate professional starts analysis in Excel—universal, flexible, familiar. But Excel breaks down with detailed assumption sets: multi-tenant lease structures, phased development projects, complex capitalization, etc. Version control becomes email chaos, formula errors compound with no audit trail, collaboration means "Final_v3_ACTUAL.xlsx" nightmares.

When financing any multi-tenant income-producing property, Altus Group's *Argus Enterprise* becomes mandatory. Lenders and conduits require it for underwriting, appraisals and loan monitoring. I used it in the 90s when it was still DOS-based. Argus delivers standardization, granularity and systematic sensitivity analysis, but at monopoly costs: $5,000+ per seat per application, mid-2000s desktop architecture wrapped in Citrix marketed as "cloud," proprietary file format locking in 30 years of data. Zero intelligence or learning—every property analyzed in isolation.

The monopoly is cracking. Altus Group's $420M Analytics segment shows 95% recurring revenue and 5-6% growth—mature monopoly, not innovator. With their September 2024 launch of *Argus Intelligence,* they basically admitted they're behind on AI (dashboards, not machine learning). January 2025's divesture of their property tax business ($700M funded buybacks, not R&D. An August 2025 press release stated that they are considering an offer to sell the company (going private).

1.1 Opportunity. Everyone—from junior analysts to senior principals—wants AI capabilities, even without fully understanding them. Firms need to signal they're using cutting-edge tools. The desktop-to-cloud transition creates switching windows. Argus's "cloud" is retrofitted legacy; Landscape would be web-native from day one. Argus Enterprise dominates income property, but Argus Developer (vertical development) and Argus EstateMaster (land development) serve a much-smaller market. We could deliver all three products' functionality for one-third the price, with AI integration and freemium accessibility for those wanting limited functionality. This isn't displacement through brute force—it's building what they can't, at a price where customers don't choose between tools.

1.2 Technical Foundation: Landscape would use modern Python-powered architecture for calculations, AI processing, and spatial analysis—enabling extensibility and user-scriptable custom analyses that desktop software can't match. React/Next.js handles the UI, PostgreSQL with PostGIS manages data and geospatial functions, and Python services power the heavy computational lifting.

2.0 Functionality. The following provides an outline of our vision for the functionality of the landscape landscape platform.

2.1 Financial Modeling. At its core, financial modeling is a sophisticated sources and uses table: how much do we spend and when, how much do we get paid and when. The math doesn't exceed undergraduate finance—DCF, NPV, IRR, equity multiples, even Monte Carlo simulations are straightforward in Excel. But organizing the inputs for any level of complexity, tracking assumptions over time with full audit trails, and running systematic sensitivity analysis is where the platform would add value. Whether analyzing a 50-unit apartment building, a 2,000-lot master-planned community, or a mixed-use project with TIF financing and multiple equity tranches, the calculation engine would need to handle it all.

Argus addresses this through three separate desktop applications: Enterprise ($5,000/year) for income-producing properties, Developer for construction projects, and EstateMaster for land development. A firm analyzing across property types pays $15,000+ per user annually for three disconnected products that can't share data or assumptions. Equity waterfalls—preferred returns, promotes, catch-ups, IRR hurdles—require an additional "Advanced Finance" module at extra cost. The products run on 1990s desktop architecture wrapped in Citrix and marketed as cloud.

Landscape would deliver all three products' functionality in a unified web-native platform for one-third the price. Single database architecture would mean assumptions flow seamlessly across property types. Full waterfall capabilities would be included, not sold as add-ons. Modern React interface with real-time collaboration, not remote desktop sessions. API-first design would mean data integrates with other tools rather than being locked in proprietary formats. Municipal bond sizing for public infrastructure financing, equity waterfall structures with unlimited tiers, scenario comparison with probability weighting—features institutional users need without paying enterprise premiums.

The complexity edge cases would emerge with unique side agreements—atypical tenant arrangements, custom equity structures, bespoke operating partnerships. This is where AI integration would become critical: interpreting deal-specific documents and generating the custom calculations required to model one-off structures that don't fit standardized templates. Python would handle the computational heavy lifting—complex time-series analysis, iterative calculations for waterfalls, sensitivity analysis across thousands of scenarios—significantly faster than JavaScript and enabling future user-scriptable extensions for custom deal structures.

2.2 Document Management System (DMS). Commercial real estate generates massive document volumes: offering memoranda, appraisals, environmental reports, surveys, title commitments, market studies, cost estimates, lease agreements, engineering plans, zoning documents, and countless internal memos and notes. Most firms store these in shared drives, Dropbox, or email attachments—creating collaboration nightmares, version control chaos, and zero institutional memory when team members leave or projects close.

We're architecting the DMS following principles established by NetDocuments, the enterprise platform serving many of the world's largest law firms. NetDocuments pioneered secure cloud-based document management with granular permissions, audit trails, and seamless integration with external systems. Landscape would adapt this institutional-grade architecture for real estate: every project would get its own document workspace with configurable folder structures, automated version control, and integration points to Outlook 365, Salesforce, and other business systems.

The core repository functions would include:

2.1.1 Version Control and Redline Management. No more "Final_v3_REVISED_USE_THIS.docx" nightmares. The system would automatically track document versions, maintain edit histories, and preserve redlined changes from Word and PDF files. Users could see what changed, when, and by whom across the entire project timeline.

2.1.2 Intelligent Organization. Folder structures would adapt to property type and deal complexity. Office acquisition projects organize differently than land development or portfolio analysis. Users could accept default structures, customize for their workflow, or create templates that propagate across similar projects.

2.1.3 Permissions and Access Control. Granular permissions would ensure internal teams, external consultants, lenders, and partners see only relevant documents. When deals close or partnerships end, access would revoke systematically rather than relying on manual cleanup.

2.1.4 External System Integration. Contact information would flow to Outlook and CRMs. Financial documents would link to models. Market data would connect to analysis tools. Python services currently fetch live market data from government sources (Census Bureau, Federal Reserve Economic Data, Bureau of Labor Statistics) ensuring demographics and economic indicators stay current. The DMS would become the central hub rather than an isolated silo.

2.1.5 Search and Retrieval. Full-text search across all project documents with filtering by date, author, document type, and custom tags. Finding the Phase II environmental report from 18 months ago would take seconds, not hours of folder diving.

Argus has no document management whatsoever. Analysts maintain their own ad hoc filing systems, often losing institutional knowledge when team members depart. Cost estimates get buried in email threads. Market studies from prior projects become inaccessible. Title commitments referencing critical easements get lost in shared drives.

The DMS would provide the infrastructure foundation. But storage alone doesn't create intelligence—that's where the AI layer would transform this repository from passive filing cabinet into active knowledge base that learns from every document uploaded.

3.0 GIS / Spatial Intelligence. Using GIS tools is not for the faint of heart. ArcGIS requires extensive training, costs thousands per seat, and demands specialists to operate. Most practitioners need spatial intelligence but can't justify tools designed for professional cartographers. They default to Google Earth.

Landscape's approach: users would never "do GIS"—they would simply see relevant spatial intelligence auto-populated. When a project is added, the platform would determine which spatial analyses matter for that property type and run them automatically using Python geospatial processing. Office gets workforce accessibility and transit proximity. Retail gets traffic patterns and trade area demographics. Industrial gets freight access and labor shed analysis. Master-planned communities get school ratings and competing project data.

3.1 Intelligent Context Awareness. When users add projects, AI would scan the knowledge base for spatially-related information. A two-year-old market study already in the system? Instant alert with contact info. Title reports often reference legal documents affecting multiple properties—special assessment districts, easements, CCRs. If Project A's title revealed a district boundary, it gets mapped. When Project B is added within the same district two years later, the user would be notified before ordering new title work.

Python geospatial libraries (GeoPandas, Shapely) would handle spatial processing. PostgreSQL with PostGIS would store spatial data alongside financial models. The open-source stack (MapLibre, PostGIS, Turf.js) could replicate 90% of proprietary GIS functionality at a fraction of ArcGIS costs, while remaining extensible for custom scripts.

Argus has zero spatial awareness—every property analyzed in isolation. CoStar provides data but no intelligence layer. Landscape would integrate spatial analysis with financial modeling automatically, making location intelligence enhance every decision without requiring GIS expertise.

4.0 AI Intelligence. This is where the platform transforms from sophisticated software into intelligent assistant. The AI layer powers all three preceding functions—extracting data from documents, generating custom calculations, determining which spatial analyses matter, learning from every project. Users could operate Landscape exactly like Argus if they choose: manually enter assumptions, ignore AI features, use it as pure calculation software. But the transformative value emerges when AI becomes the brain making the DMS, financial models, and GIS work together intelligently.

4.1 Document Intelligence & Data Extraction. When users upload documents—offering memoranda, market studies, cost estimates, environmental reports, lease agreements—AI would extract structured data and populate model assumptions automatically. Python NLP services would parse for lease terms, financial assumptions, cost estimates, market benchmarks, and risk factors. But extraction is just the start. The AI would validate: Environmental report mentions contamination but recommends no action? "Document acknowledges risk but provides no

remediation plan—consider third-party review." User underwrites expenses at 25% when market studies show 30-35%? "Operating expenses appear optimistic compared to benchmarks."

Every contractor proposal would build institutional memory. Roof replacement estimate for 50,000 SF? AI extracts unit costs, adjusts for location. Two years later, new project needs roof work: "We have cost data from similar scope—estimated $X based on adjusted factors."

4.2 Project-Specific AI Assistant. Each project would have its own AI chat assistant with full context awareness—query documents ("What did Phase II say about contamination?"), validate assumptions ("Does rent growth align with market study?"), generate narratives ("Write executive summary for lender"), identify risks, trace decisions back to source documents, and flag gaps in diligence packages. Cross-project intelligence would query across boundaries: "Show absorption assumptions from our last 5 Phoenix multifamily deals."

4.3 Narrative Generation & Market Intelligence. AI would generate investment memos, executive summaries, and diligence reports grounded in actual project data—not generic boilerplate. Automatic progress reporting: weekly emails showing documents added, assumptions changed, risks emerged. Consistency checking: flag anywhere narrative conflicts with model assumptions. Multiple broker studies uploaded? AI creates unified competitive landscape showing consensus vs. outliers. Three appraisals cluster at 6.5-6.8% cap rates, one outlier at 7.2%? AI synthesizes the differences and explains why.

4.4 Outcome Learning & Spatial Intelligence. The platform would track projected vs. actual performance, identifying forecast error patterns using Python ML models: "Your office acquisitions consistently underwrite rent growth at 3% but actual averages 2.1%—consider adjusting." "Phoenix multifamily with light rail access lease up 20% faster than your assumptions." For spatial intelligence, AI would determine which analyses matter for each property type and run them automatically, learning from portfolio performance: "Your highest-performing office assets average 0.3 miles from light rail" or "Subdivisions in 8+ rated school districts achieve 25% premium in your deals."

4. The Compound Intelligence Effect. Every document uploaded teaches the system. Every cost estimate builds the database. Every market study refines benchmarks. Every project adds spatial intelligence. The fiftieth project would be smarter than the first because the system remembers everything from the previous forty-nine. This network effect creates defensible competitive moats—the platform becomes increasingly valuable the longer firms use it.

Firms embracing the AI layer would see 90% reduction in model setup time, institutional memory surviving team turnover, risk identification before problems emerge, benchmarking against historical performance, and continuous forecast accuracy improvement. Argus has zero learning capability—every project starts from scratch, analysts manually recreate work validated on previous deals. With Landscape, intelligence compounds, becoming the firm's persistent competitive advantage.

5.0 Development Status - Foundational Infrastructure. Over the last two months, with AI assistance (Claude and ChatGPT), I've built what both insist is a solid and scalable foundation. The architecture uses modern web stack: React/Next.js 15 frontend, PostgreSQL with PostGIS for spatial data, Python services for data engineering, and MeiliSearch for document indexing. The database schema (`landscape` schema, 74 tables) is normalized and extensible, designed to handle everything from simple office buildings to complex master-planned communities with thousands of lots.

5.1 Planning Hierarchy & Data Model: The universal hierarchy system works—configurable 4-level structure where level labels adapt to property type (e.g., "Area/Phase/Parcel" for land development, different labels for commercial properties). Built using normalized container system (`tbl_container` with parent-child relationships) plus `tbl_project_config` for custom level labels. Land use taxonomy is fully database-driven: 9 families (Residential, Commercial, Industrial, Office, Multi-Family, Hospitality, Institutional, Mixed Use, Open Space), cascading types within families, product catalogs with lot dimensions. Users can create projects, define planning structures, assign land uses, track acreage and units. The UI includes tile-based grid layouts with inline editing, drag-and-drop organization, and real-time updates via SWR data fetching.

5.2 Document Management System: The DMS foundation exists following NetDocuments architectural principles—document registry (`core_doc`), template system (`dms_templates`, `dms_attributes`), workspace organization, version control tracking, and full-text search via MeiliSearch. File upload infrastructure works (UploadThing integration). The database can store document metadata, extraction queues, and profile attributes. What's missing: the AI intelligence layer that makes documents useful. Documents upload and organize, but they don't extract data, populate assumptions, or build institutional memory yet.

5.3 GIS Infrastructure: PostgreSQL with PostGIS extension installed, boundary tables created (`gis_project_boundary`, `gis_plan_parcel`, `gis_tax_parcel_ref`), and basic MapLibre visualization working. Python geospatial libraries integrated (GeoPandas, Shapely ready to use). The database can store geometries and spatial relationships. What's missing: automated spatial analysis, intelligent context awareness, property-type-specific heuristics. Right now it's a map viewer, not spatial intelligence.

5.4 Market Data Foundation: Python market ingestion engine operational—CLI tool that fetches demographic and economic data from Census Bureau (ACS), Federal Reserve (FRED), Bureau of Labor Statistics (BLS), and Federal Housing Finance Agency (FHFA) APIs. Data populates `market_data_series` tables with location-based time series. The infrastructure for growth rate assumptions exists (`core_fin_growth_rate_sets`, `core_fin_growth_rate_steps`) with UI for creating absorption/pricing/cost escalation curves. What's missing: AI validation of user assumptions against market data, anomaly detection, cross-project pattern recognition.

5.5 Budget System Structure: Financial data model built—hierarchical category system (`core_fin_category` with 4 levels), units of measure (`core_fin_uom`), budget fact tables (`core_fin_fact_budget`, `core_fin_fact_actual`), funding sources, and tag systems. Two budget grid UIs exist: MUI DataGrid (light theme) and Handsontable (dark theme, spreadsheet-

like). Users can enter line items with quantities, unit costs, categories, and tags. Grid functionality works: sorting, filtering, inline editing, CRUD operations.

6.0 Development Status – What's Missing.

6.1 No Financial Calculations: This is the critical gap. The budget grids sum columns, but there are no cash flow projections, no IRR/NPV calculations, no waterfall distributions, no sensitivity analysis, no scenario modeling. The data structure is ready—periods defined, funding sources configured, equity structures modeled—but the Python calculation engine doesn't exist. The math itself is undergraduate finance (DCF, IRR, equity multiples), but I cannot assess:

- Performance requirements at scale (1,000-period monthly cash flows, 50-tranche waterfall, 100-scenario Monte Carlo)
- Whether web architecture handles calculation intensity or needs dedicated compute
- How to structure the calculation pipeline (real-time vs. queued jobs vs. cached results)

6.2 No Production AI Features: The document extraction prototype works—upload PDF, Claude Vision API analyzes pages, extracts structured fields, returns confidence scores—but it's a demo, not production system. Missing:

- Batch processing pipeline for high-volume document ingestion
- Training data and accuracy validation for CRE-specific documents (offering memos, appraisals, PADs)
- Cost estimation database that builds from contractor proposals
- Contact extraction and relationship mapping
- Intelligent tagging and organization that learns user preferences
- Market intelligence synthesis from multiple conflicting sources
- Outcome learning that improves forecast accuracy over time

6.3 No Spatial Intelligence: The GIS displays boundaries and parcels, but doesn't analyze anything. The property-type heuristics we envision—office workforce accessibility scoring, retail trade area demographics, industrial labor shed analysis, residential school district ratings—none of that logic exists. The pattern recognition from portfolio performance (learning that office properties near transit outperform, or subdivisions in top-rated school districts command premiums) requires ML models we haven't built.

6.4 No User Authentication: Zero security. No login, no roles, no permissions, no multi-tenancy. Every user sees everything. This is acceptable for prototype but obviously unsuitable for production.

7. Core Admission. I know the CRE market problem intimately. I understand the calculation engine requirements because I've built these models in Excel for 35 years. I've architected the data model based on decades of domain knowledge. What I don't know—and cannot assess on my own—is *what AI can do today* vs. what requires custom ML development vs. what's 2026 capability not ready for 2025 deployment.

7.1 Feasibility Questions. The following are several feasibility questions that I need to understand before I could propose any capstone project.

1. **Document parsing accuracy**: Is extracting lease terms, cost estimates, and market assumptions from PDFs a commodity problem (use existing APIs) or does it require custom training data? What accuracy rates are realistic for CRE documents? How much training data would be needed to get from 70% accuracy to 95%?
2. **Market intelligence & anomaly detection**: Can we bootstrap predictive models without years of proprietary data, or do we need massive datasets first? What's state-of-the-art for automated comparable analysis? Can we use public data (CoStar, REIT disclosures) to train initial models?
3. **Outcome learning**: Can this work without years of proprietary outcome data, or is this a "build audience first, add intelligence later" feature? How do platforms like this achieve learning effects in early stages?
4. **Spatial pattern recognition**: Is standard ML (clustering, correlation analysis) sufficient for identifying spatial performance patterns, or do we need custom models? What data volumes are required for meaningful insights?
5. **Agent architecture**: For multi-agent collaboration (lease abstraction agent → market validation agent → financial modeling agent), is this production-ready or experimental? What frameworks exist? What's realistic for 2025?
6. **Calculation engine architecture**: Given undergraduate-level math but high iteration counts, do we need anything beyond Python + pandas + numpy, or are there architectural considerations we're missing? What's the right pattern for real-time calculations vs. background processing?