Understanding AI Technology for Business Problems Final Project
Executive Assistant for MBA Students (MBAEA)

Divya Bade divbade@stanford.edu; Nikhil Gupta nikhilgu@stanford.edu; Lingu Zhong zhonglg@stanford.edu; Tanvi Sharma tanvi19@stanford.edu; Juan C. Barriga jcbarrig@stanford.edu

A. Business Problem

As MBA students, we have a perpetual question: **Am I making the most of my MBA program?**

We are paying high tuition and feel the pressure to reap bang for our buck. Our time is spread incredibly thin between professional goals, hobbies, social networking commitments, family time, and the list goes on. We are fed information and make decisions across a number of platforms: Gmail (~50 GSB Blast emails/day), Partiful, Luma, iMessage, WhatsApp, Canvas (many deadlines), and our personal note-taking platforms like Notion, Zoom, and Granola. Thus, figuring out how to be productive is exhausting.

We are building an "executive assistant-like" agent that helps **MBA students (target customer segment)** like us **make time (plan, schedule)** for **what matters the most (prioritize)** to us as **unique individuals (based on our personal preferences)**.

B. AI/ML application

MBAEA helps students make the most of their time by: (1) understanding their goals and underlying priorities; (2) retrieving data from relevant sources such as Canvas, Partiful, and messaging platforms; (3) breaking goals into bite-sized tasks; (4) scheduling those tasks on their calendar; (5) providing daily recaps of time spent relative to goals; and (6) capturing feedback to continuously refine recommendations. The system is built on a combination of AI/ML techniques that work in concert.

MBAEA's architecture moves beyond generic chat flows by centering "intelligence" in a specialized intent model and a constraint-based planning engine. We use a three-stage alignment process to transform fragmented signals into a personalized schedule.

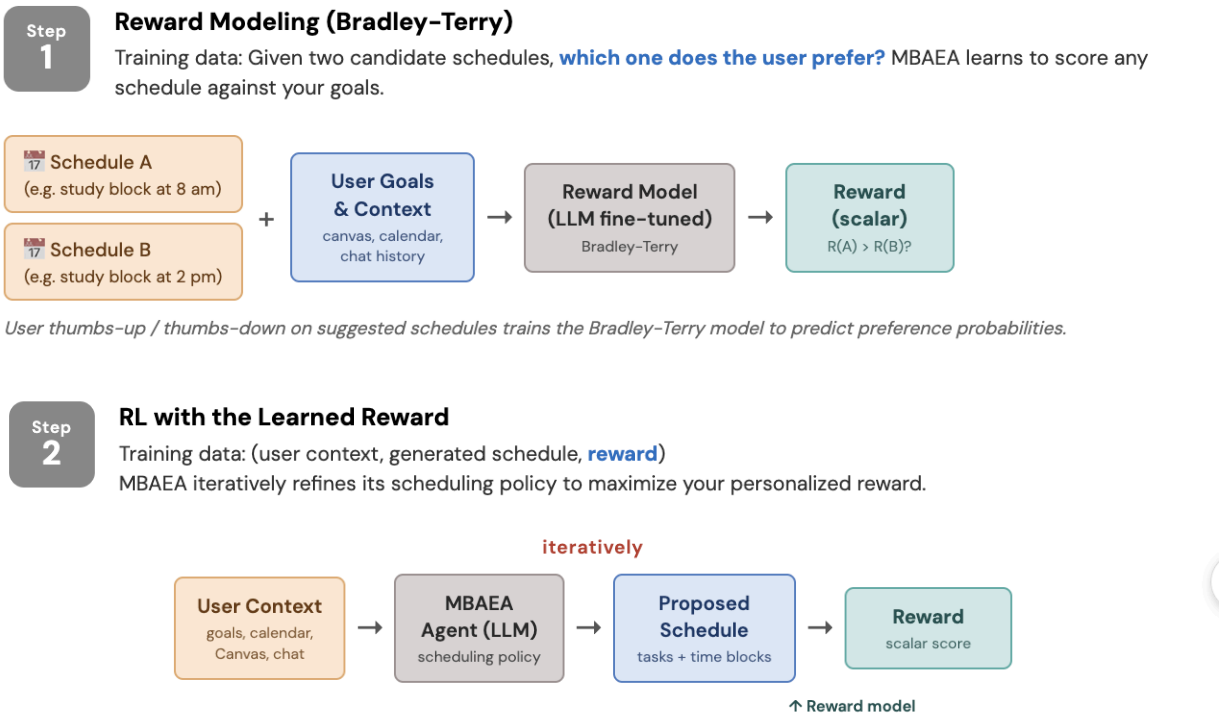
1. Structured Extraction (Supervised Fine-Tuning)

The first stage is structured extraction: the system ingests unstructured data from sources like WhatsApp, Canvas, and Gmail, using a supervised fine-tuned LLM as a structured extractor. The model is tuned to maximize the probability of turning messy human inputs into standardized "candidate commitments," extracting specific deadlines, estimated durations, and required actions. This ensures downstream components operate on consistent, high-quality data rather than unpredictable free text. Concretely, we fine-tune a small instruction-tuned LLM on labeled

examples of raw message snippets mapped to structured outputs (task name, deadline, estimated duration, and priority signal). We apply supervised fine-tuning with loss computed over the output tokens, so the model learns the extraction task specifically rather than general language modeling. MBAEA also handles multimodal inputs: a voice memo, for instance, can be transcribed.

2. Intent and Reward Model (Personalization)

On top of the extraction layer, MBAEA trains a model to predict the "strength" and "goal alignment" of every commitment (e.g., distinguishing a social from a hard recruiting deadline). To achieve deep personalization and solve the cold-start problem, we use three distinct learning phases. First, structured onboarding requires users to define core goals (e.g., prioritizing academics over social), providing the initial human-written demonstrations needed to seed the model. Next, we employ weak supervision, using platform-specific cues (like due dates in Canvas) to automatically label tasks before receiving direct user feedback. Our labeling heuristics include: Canvas items due within 72 hours labeled high-priority; external-participant calendar events ranked above internal ones; and messages containing urgency keywords. These pre-train the reward model before any real user feedback is available. Third, we utilize reinforcement learning from human feedback. As users interact with the app through thumbs-up or thumbs-down signals between proposed schedule pieces, the system treats these actions as a reward signal. By applying a Bradley-Terry model, MBAEA learns to rank proposed schedule chunks based on what the user actually values, effectively building a mathematical reward model of the student's unique preferences.



3. Optimization and Constraint-Based Planning

Finally, an optimization layer turns these predicted rewards into an actionable schedule. The system operates as an agentic AI: it reasons about competing priorities, constructs multi-step plans, takes direct actions like scheduling calendar blocks, and adapts dynamically as goals evolve. While the LLM understands intent, the planning engine handles the "physics" of time, treating scheduling as a constraint-satisfaction problem that allocates focus blocks and tasks into available slots while strictly respecting constraints (no overlaps, sleep windows, and maximum daily commitments). The engine maximizes an objective function that seeks to maximize "goal progress" (a weighted sum of reward model scores across scheduled tasks) while minimizing two penalty terms: "context switching" (between unrelated goal categories) and "overload" (beyond a soft hour cap). Hard constraints (no overlaps, fixed sleep windows, etc.) are enforced separately as boundaries the optimizer cannot trade off against.

A standard LLM alone would be insufficient for this problem. Prompting an off-the-shelf model to "suggest how to spend my day" produces generic recommendations without mechanisms to learn preferences over time, enforce hard constraints like calendar conflicts, or take autonomous action. MBAEA's three-stage architecture addresses each gap: supervised fine-tuning ensures reliable extraction from noisy inputs; the reward model builds a mathematically grounded representation of what each student actually values; and the constraint-based planner guarantees recommendations are feasible and automatically implemented. Together, these layers produce a system that compounds in usefulness the longer it is used.

C. Data Collection

To build MBAEA, our system integrates multiple types of behavioral, contextual, and preference data to deliver personalized scheduling, prioritization, and productivity recommendations. The key categories of required data and collection approaches are outlined below.

1. User goals, preferences, and priorities: The system requires data on users' goals (academic, professional, and personal goals), along with their preferences for time allocation, event participation, and productivity habits. This information is collected through onboarding questionnaires, conversational chatbot interactions (WhatsApp), and ongoing feedback on recommendations.

2. Scheduling, event, and deadline data: To help users plan their time effectively, the system aggregates data from the aforementioned scheduling, communications, and event sources. Data collection can occur via API or inbox/text messages parsing, and optional manual inputs. This aggregation enables the assistant to provide consolidated scheduling recommendations and prevent missed deadlines or conflicting commitments.

3. Behavioral and activity data: Understanding how users actually spend their time is critical for personalized recommendations. Relevant data includes meeting attendance, time spent on

tasks, engagement levels, and productivity patterns. This can be collected through calendar confirmations, chatbot check-ins, and integrations with productivity tools such as note-taking or meeting platforms (e.g., accepted invites with confirmation of attendance). Such data enables features like daily recaps and adaptive scheduling recommendations.

4. Communication and multimodal inputs: Since the assistant interacts through conversational interfaces, it must process text messages, voice notes, and shared documents. Voice inputs would require speech-to-text processing, while natural language processing would help extract actionable tasks, deadlines, or priorities from conversational data.

Challenges in Data Collection and Preparation

1. Data fragmentation and inconsistency: Relevant data is spread across multiple platforms and formats, from structured calendar entries to unstructured WhatsApp threads and voice memos. Each source uses different schemas, terminology, and update frequencies, making it non-trivial to reconcile them into a unified representation. For example, a Canvas deadline and a casually mentioned "let's meet Friday" in iMessage must both be parsed and normalized into the same commitment format. Building reliable ingestion pipelines with consistent entity resolution (e.g., de-duplicating the same event mentioned across email, Partiful, and text) and robust error handling for API rate limits or outages are essential infrastructure work.

2. Cold-start personalization: Early recommendations may be less accurate due to limited user data. Structured onboarding, designed to be thorough enough to seed the model but short enough not to deter adoption, provides an initial preference signal. However, it cannot fully capture the nuance of individual behavior. To bridge this gap, MBAEA uses weak supervision from platform-specific cues (e.g., Canvas due dates as proxy signals for high-priority tasks) before direct feedback is available. Gradual behavioral learning through accepted or rejected recommendations then refines the model over time, but this means the first couple of weeks of use may feel less tailored, creating a critical window where early user experience must be managed carefully to avoid churn.

3. Multimodal processing complexity: Handling voice, text, and documents requires robust NLP and speech processing systems to ensure accurate understanding. Voice memos introduce transcription errors that can cascade into incorrectly extracted deadlines or misclassified priorities. Informal language – abbreviations, sarcasm, or incomplete sentences – further complicates intent extraction. Documents shared via email or Canvas may be PDFs or slides requiring additional parsing layers. Each modality demands its own preprocessing pipeline, and errors compound across stages: a mis-transcribed date in a voice note that then gets scheduled incorrectly is a high-friction failure that erodes user trust quickly.

D. Solution Evaluation

We will evaluate the effectiveness of MBAEA by testing whether it helps MBA students allocate their limited time toward what matters most to them, through reducing coordination and planning burden and maintaining commitment to their stated goals.

Our evaluation follows the logic chain: 1) MBAEA correctly understands the context and user intent, and can translate it into high-quality action suggestions; 2) the user will accept and execute those recommendations (we can track this through accepted calendar changes, task completion, or periodic prompting for thumbs up/down on whether recommendations are actually useful); 3) over time, users should see reduced "MBA overhead" and improvements in personal goal progression.

We will run a short baseline period (~2 weeks) where users track their time allocation and priorities without MBAEA, and then an intervention period (~4 weeks) with MBAEA. We will also introduce 2 cohorts with different onboarding time to reduce seasonality effects.

Success Metrics	Measurement	Target
Recommendation acceptance rate	Accepted recommendations / Total suggestions	>60%
Follow-through rate for high-priority goals	Completed high-priority tasks / scheduled high-priority tasks	>70%
Agentic task completion volume	# agent-completed tasks (e.g., book a meeting, move a calendar invite) per user per week	>15
4-week retention	Active users in week 4 / Week 0 Cohort	>60%

Unlike general-purpose LLM assistants or desktop-bound tools like Claude Cowork, MBAEA is purpose-built for the MBA context across four dimensions: cross-platform API ingestion that captures signals across apps rather than local files only; a trained reward model that compounds personalization over time through RLHF rather than relying on a stateless LLM; a WhatsApp-native interface that meets students where they already are; and a server-side, always-on architecture that delivers timely nudges and uninterrupted scheduling regardless of whether a desktop app is open.

E. Pitfalls and Ethical Considerations

A core risk with MBAEA is hallucinating or making mistakes in a domain that is highly subjective: how a student should spend their time. Because the agent is designed to synthesize

inputs across classes, recruiting, social commitments, and goals, and then break goals into tasks and proactively recommend or schedule actions, it can confidently push a plan that is misaligned with what the user actually values. Cold-start personalization, noisy/ambiguous inputs (especially from unstructured text), and imperfect prioritization can lead to high-friction failure modes (e.g., missed deadlines, calendar conflicts, “phantom” commitments inferred from casual messages, and unhelpful nudges). There is also a dependency risk: if users begin to outsource planning and reflection to the tool, errors can feel higher stakes, and users may gradually rely on the assistant’s recommendations instead of building their own judgment and routines.

Ethically, the biggest issues are data privacy and consent. MBAEA’s value proposition depends on aggregating sensitive, cross-platform information (calendar/email, academic context, and potentially messaging or voice notes). That creates risks of over-collection, retention of sensitive data longer than necessary, accidental exposure through logs or vendor integrations, and leakage of third-party information contained in messages or invites. Canvas integration specifically may implicate FERPA, which restricts how student educational records may be shared and processed by third parties. In addition, there are bias risks in both interpretation and recommendations: the system can learn skewed notions of what is “high value” (e.g., over-weighting certain event types, communication styles, or goals) and repeatedly reinforce those patterns through the thumbs up/down feedback loops. For instance, someone stressing over building a more robust community on campus might obsess over packing their calendar with social commitments, leading to even more stress from “over-optimizing.”

Mitigations should emphasize explicit user controls over which sources are connected and what actions are allowed. One example is requiring confirmation before any calendar block is created, not scheduling autonomously by default. Strict data minimization and retention policies should limit what is stored and for how long, with sensitive message content that is not persisted in logs. Transparency into why a recommendation was made, such as "scheduled based on your recruiting priority and a 3-hour deadline buffer," builds trust and helps users catch misalignments early. Clear fallback behaviors when confidence is low are equally important: rather than committing to a single directive, MBAEA should present ranked options with uncertainty flagged, letting the user remain the final decision-maker. Audit logs of agent-taken actions (e.g., moved calendar event X) should be easily accessible and reversible, ensuring users never feel the system has acted beyond their boundaries. A related concern is third-party data: when MBAEA ingests a WhatsApp thread or email chain, it processes personal information about other students who never consented to the system. We can process message content ephemerally (extracting the time, place, and action, and immediately discarding the raw text), and surfacing a clear disclosure to users that their integrations may capture information about others.

F. Trade-offs in Deployment:

Implementing MBAEA involves navigating significant trade-offs between system performance, user experience, and operational costs. A primary trade-off exists between accuracy and latency;

utilizing frontier LLMs like GPT-5 for real-time extraction from messy WhatsApp or voice notes ensures high fidelity but introduces noticeable delays and higher API costs. To mitigate this, we may deploy smaller, faster models for routine extraction while reserving more advanced models for complex weekly recaps. Furthermore, there is tension between personalization and scalability. While the constraint-based planning engine provides highly tailored schedules, the computational intensity of solving complex optimization problems for thousands of students simultaneously could challenge system responsiveness as the user base grows. Lastly, we face a privacy-utility trade-off. To provide the most "executive" level of assistance, the agent requires deep access to private communications; however, implementing the strict data-masking and local-processing protocols necessary for user trust may limit the agent's ability to cross-reference subtle social nuances or private commitments, thereby reducing its overall helpfulness.

G. Limitations and Future Directions

The current scope of MBAEA is subject to several key limitations, most notably the lack of emotional and physiological context. While the system understands what is on a calendar or what information a user actively provides, it cannot yet perceive a student's fluctuating energy levels or burnout in real life, potentially leading to overload during periods of high stress. Additionally, the system is currently designed for individual optimization, ignoring the social coordination inherent in the MBA experience where schedules are often codependent on study groups or project teams.

Future iterations will incorporate biometric integration (e.g., via Oura or Apple Watch) to refine energy-aware scheduling, automatically shifting high-cognition tasks to focus blocks when a user's physiological data indicates peak alertness. This can also be leveraged to roll out an "SOS mode" feature that more heavily encourages a user to cancel on optional commitments when they are unwell. We also plan to move from a reactive feedback loop to proactive goal discovery, using long-term behavioral trends to suggest new, high-alignment goals that the student may not have explicitly codified during onboarding. Finally, we will explore multi-agent coordination, enabling individual MBAEAs to "negotiate" with one another to find optimal meeting windows without manual intervention from the users.

H. Prototype

Prototype link used for demos - *create an account with any email address!*