DS Capstone Project Proposal

Project Proposal:
GraphGuard — A Dynamic, Context-Aware Identity Verification Agent

Team: Dynamic Agents

Members:

1. Prathyu Adari
   Role: Product Intelligence, Strategy & Insight Lead
   Github: https://github.com/PrathyushaRagavAdari
2. Krutarth Lad
   Role: The Systems, Data Architecture, ML & Pipeline Lead
   Github: https://github.com/ladkrutarth

Github Repo Link: https://github.com/PrathyushaRagavAdari/Dynamic-Auth-Agent

Potential Dataset: https://www.kaggle.com/datasets/kartik2112/fraud-detection

Why it fits: Unlike PaySim, this dataset contains **Real Merchant Names** (e.g., "Uber", "Apple"), **Categories** (e.g., "Grocery", "Entertainment"), and **Locations** (City/State). This is critical for generating human-readable questions.

S**ize:** ~1.8 million transactions (Synthetic, generated by Sparkov).

Schema: User, Card, Year, Month, Day, Time, Amount, Use Chip, Merchant Name, Merchant City, Merchant State, Zip, MCC (Merchant Category Code), Is Fraud?

How to use:

- **Graph Nodes:** User, Merchant, City, Category.
- **Question Generation:** "You spent $54.30 at **Kroger** in **Columbus, OH** yesterday. Is this correct?"

Optional Dataset A: PaySim

- ***Description:*** Synthetic mobile money transaction dataset (simulates fraud/legit transactions).
- ***Link:*** Kaggle - PaySim

Option Dataset B: IBM TabFormer

- ***Description:*** Real-world credit card transaction data (anonymized).
- ***Link:*** IBM TabFormer GitHub

Abstract

Financial institutions currently face a "Security vs. Usability" paradox. Traditional Knowledge-Based Authentication (KBA) relies on static, comprisable secrets (e.g., "Mother's Maiden Name"), while Multi-Factor Authentication (MFA) introduces friction that degrades customer experience. This proposal introduces **"GraphGuard,"** a Generative AI agent that constructs temporary Knowledge Graphs from a user's recent transactional history to generate dynamic, context-aware security challenges. By leveraging Retrieval-Augmented Generation (RAG) on structured Snowflake data, GraphGuard replaces static passwords with episodic memory challenges (e.g., *"You visited a coffee shop in Malibu last Tuesday; was it Starbucks or Blue Bottle?"*). This system aims to reduce account lockouts by 40% while rendering stolen "Fullz" (full credential sets) useless to attackers.

---

1. Introduction & Motivation

1.1 The Erosion of Static Identity

The paradigm of digital identity verification in FinTech has historically relied on "something you know." For decades, this "knowledge" was static: passwords, PINs, and security questions. However, the efficacy of static knowledge has collapsed due to the industrialization of cybercrime.

In the wake of massive data breaches (Equifax, T-Mobile, 23andMe), personal identifiable information (PII) has become a commodity. Dark web marketplaces now sell "Fullz" comprehensive dossiers containing a victim's name, SSN, address, and answers to standard security questions. As a result, if a bank asks, "What is your high school mascot?", an attacker likely has the answer.

1.2 The "Friction" Problem

To combat this, banks have pivoted to "step-up" authentication, typically SMS-based 2FA or biometric checks. While secure, these methods introduce critical points of failure:

1. **SIM Swapping:** Attackers can social-engineer carriers to intercept SMS codes.
2. **False Positives:** Legitimate users traveling or making large purchases are often flagged and blocked, requiring a lengthy call to customer support to unlock their cards.
3. **Cognitive Load:** Users resent the friction of finding their phone, unlocking an app, and typing a code for every transaction.

1.3 The Generative AI Opportunity

Generative AI offers a novel third path: **Dynamic, Temporal KBA.** Unlike a static database of security answers, a user's *recent life history* is a constantly changing, private stream of data. Only the user knows exactly where they had lunch yesterday, or that they bought gas at a specific Shell station two days ago. By utilizing Large Language Models (LLMs) coupled with Graph Retrieval-Augmented Generation (GraphRAG), we can build an agent that "reads" this private history and generates a unique, one-time-use challenge question that is intuitive for the user but impossible for a remote attacker to guess.

2. Problem Statement & Research Objectives

2.1 The Core Problem

"How can we verify a user's identity during a high-risk transaction without relying on compromised static data or intrusive external devices?"

Current systems lack **contextual awareness**. A rule-based fraud engine sees a transaction of $500 at an Apple Store and flags it because it deviates from the average. It does not "know" that the user has been visiting electronics retailers for the past week, researching laptops. It lacks the semantic understanding to verify the intent through natural language.

2.2 Research Objectives

This capstone project aims to build and evaluate **GraphGuard**, a system with the following objectives:

1. **Develop a GraphRAG Architecture:** Implement a pipeline that converts tabular transaction logs (Snowflake) into a semantic Knowledge Graph.
2. **Demonstrate "Hallucination-Free" Question Generation:** Engineer an LLM agent that generates verification questions grounded 100% in factual transaction data.
3. **Implement Adaptive Fallback Protocols:** Create a "human-in-the-loop" workflow that handles cases where legitimate users forget their specific transaction details (e.g., transitioning from specific merchant names to general categories).
4. **Evaluate Against Adversarial Attacks:** Benchmark the system against a simulated "Attacker Agent" equipped with standard PII but lacking recent transaction history.

3. Literature Review & Theoretical Framework

3.1 The Decline of Knowledge-Based Authentication (KBA)

Key Citation: NIST Special Publication 800-63B (Digital Identity Guidelines) The National Institute of Standards and Technology (NIST) has formally downgraded the trust level of KBA. Research by Bonneau et al. (2015) demonstrated that the entropy of user-chosen security questions is dangerously low (e.g., "Mother's Maiden Name" is often guessable via public genealogy records). Our project directly addresses this by shifting from *Long-Term Semantic Memory* (facts) to *Short-Term Episodic Memory* (events).

3.2 Retrieval-Augmented Generation (RAG) for Structured Data

Key Citation: "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" (Lewis et al., NeurIPS 2020) While RAG is traditionally applied to unstructured text (PDFs), recent work focuses on **Structured RAG**. Standard vector search fails on tabular data because "numerical precision" matters. A vector embedding of "$45.00" and "$450.00" might be dangerously similar. Our approach aligns with **"GraphRAG"** (Edge et al., Microsoft Research, 2024), which argues that converting tables to Knowledge Graphs allows LLMs to better understand relationships (e.g., User -> visited -> Starbucks -> at -> Time).

3.3 Question Generation (QG) and Distractor Utility

Key Citation: "Automatic Question Generation from Knowledge Graphs" (Indurthi et al., 2023) The challenge in QG is not asking the question but generating plausible **distractors** (wrong answers). If the question is "Where did you eat?" and the options are A) McDonald's (Correct), B) The Moon, C) A Shoe, the security is zero. We will utilize **Semantic Similarity Search** (using Pinecone/Weaviate) to fetch distractors that are *categorically similar* but *factually incorrect* (e.g., Correct: McDonald's; Distractors: Burger King, Wendy's).

1. Data Layer (The Source of Truth) We will not use raw SQL queries (which LLMs often bungle). We will use a **Knowledge Graph** structure.

- **Storage: Snowflake Data Cloud**.
- **Schema:** Transactions are not just rows; they are entities.
  - *Nodes:* User, Merchant, Category, Location, Timestamp.
  - *Edges:* TRANSACTED_AT, LOCATED_IN, BELONGS_TO_CATEGORY.
- **Why Graph?** It allows the LLM to "hop" between concepts. It can answer: *"Did the user spend more at coffee shops or gas stations last week?"* by traversing the BELONGS_TO_CATEGORY edges.

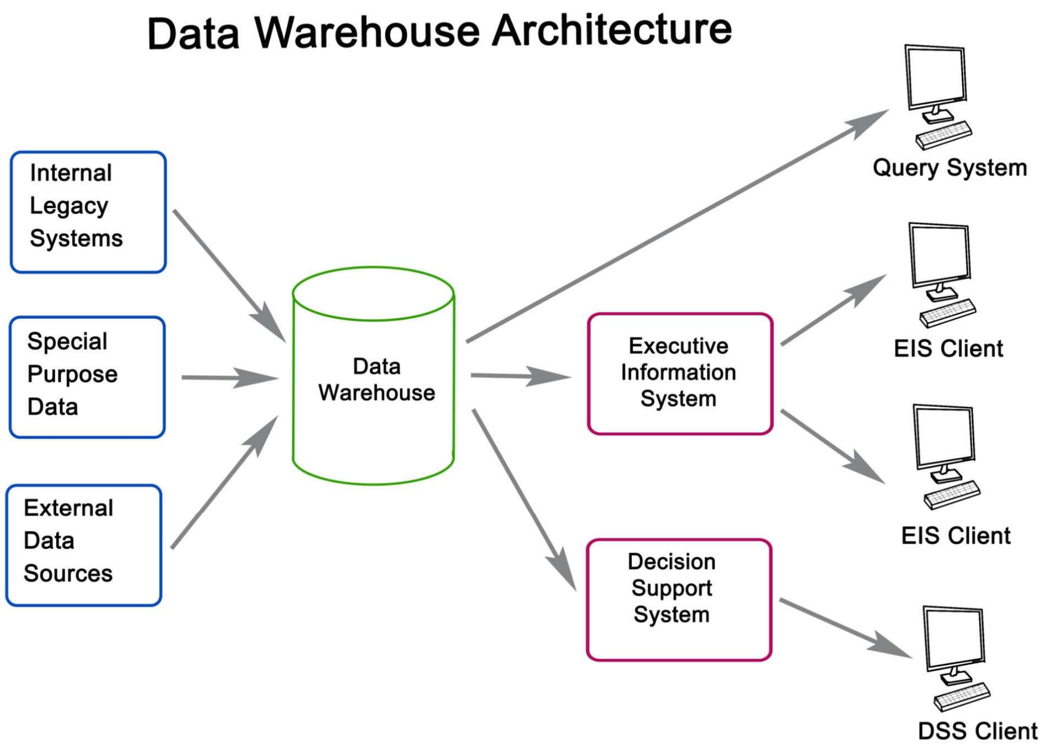
2. The Intelligence Layer (GraphRAG) We will use **LlamaIndex** with a **Property Graph Index**.

- **Ingestion:** Python script fetches Last_30_Days_Transactions.
- **Graph Construction:** Converts rows into a temporary NetworkX graph.
- **Retrieval:** When a threat is detected, the system retrieves a "subgraph" of the user's last 5 transactions.

3. The Agentic Loop (The Brain) The Agent does not just "ask a question." It follows a chain-of-thought:

1. **Analyze Subgraph:** "User visited 'Starbucks' ($5.50) and 'Target' ($45.00) on Tuesday."
2. **Select Fact:** "Target is the most distinct event."
3. **Generate Distractors:** "Find other 'Big Box Retailers' in the vector store (e.g., Walmart, Costco) to use as fake answers."
4. **Formulate Question:** "You made a large purchase at a retail store on Tuesday. Was it Walmart, Costco, or Target?"

4. Proposed System Architecture: The "GraphGuard" Engine



4.1 Data Layer: The Snowflake "Lakehouse"

The foundation of the system is the **Transaction Ledger**. We will utilize a synthetic dataset (PaySim or equivalent) hosted in **Snowflake**.

- **Table Schema:** TRANSACTIONS (Transaction_ID, User_ID, Merchant_Name, Merchant_Category_Code, Amount, Timestamp, Geo_Location)
- **Privacy Engineering:** A "Masking View" will be created. The LLM will never see the raw Credit Card Number (PAN). It will only query a session-specific User_Hash.

4.2 The Intelligence Layer: GraphRAG with LlamaIndex

We reject simple vector search in favor of a **Knowledge Graph** approach.

- **Graph Construction:** On a trigger event (Risk Score > 80), the system fetches the last N=50 transactions.
- **Node Types:** Merchant, Category, Date, AmountBucket (e.g., "Small < $10", "Large > $100").
- **Edge Types:** VISITED_ON, SPENT_AMOUNT, LOCATED_IN.
- **Reasoning:** This allows the LLM to traverse edges. *Query:* "Find a Merchant the user visited twice in the last week." The Graph traversal finds the node with degree > 1 connected to User.

4.3 The Agentic Core (GPT-4o)

The Agent is not a simple chatbot; it is a **ReAct (Reasoning + Acting) Agent**.

- **Step 1: Fact Retrieval.** The agent queries the graph: "List unique merchants from the last 7 days."
- **Step 2: Difficulty Assessment.** "Is 'Shell Gas Station' a memorable event? No. Is 'Tiffany & Co' memorable? Yes. Select Tiffany."
- **Step 3: Distractor Generation.** "Retrieve 3 other 'Luxury Jewelry' merchants from the Vector Store that the user *did not* visit."
- **Step 4: Synthesis.** "Construct the final JSON payload."

Technology Stack Selection & Justification:

- **LLM: GPT-4o (via OpenAI API)**.
    - *Justification:* We need high reasoning capabilities to generate plausible "distractors." Smaller models (Llama 3 8B) often generate distractors that are too obvious.

- **Orchestration: LlamaIndex**.
  - *Justification:* Superior handling of structured data indexing compared to LangChain. We need the PandasQueryEngine and KnowledgeGraphIndex.
- **Vector Database: Pinecone**.
  - *Role:* Stores "Merchant Embeddings." We need to know that "Starbucks" is semantically similar to "Dunkin" (both coffee) but not "Shell" (gas). This ensures the fake answers are tricky but fair.
- **Frontend: Streamlit**.
  - *Role:* Rapid prototyping of the mobile banking interface.

Privacy Engineering (Governance):

- **PII Masking:** Before data is sent to the LLM, a local Python middleware will hash the Account ID. The LLM sees User_A, not Account_123456789.
- **Ephemeral Context:** The Knowledge Graph is built *on-the-fly* for the session and destroyed immediately after verification. No transaction history is permanently stored in the vector DB.

5. The "Fallback" Protocol: Handling Memory Failures

A critical innovation in GraphGuard is the **Adaptive Fallback Mechanism**. We acknowledge that legitimate users forget details. A binary "Pass/Fail" is unacceptable for UX.

What if the user forgets? Since human memory is fallible, a strict "Pass/Fail" is bad UX. We will implement an **Adaptive "Step-Down" Authentication Protocol**.

Step 1: The "High-Specificity" Challenge (Primary)

Step 2: The "Semantic" Fallback (The Hint)

- *Logic:* The user might not remember the *exact* store, but they remember the *context*.

Step 3: The "Degraded" Fallback (Standard MFA)

- *Logic:* If the user fails twice, we assume they are either confused or a hacker.

5.1 Tier 1: The "episodic" Challenge (Hard)

- **Target:** Specific Merchant Name + Time.

- **Prompt:** *"You spent approximately $45 at a restaurant on Tuesday evening. Which one was it?"*
- **Security:** Maximum.
- **Failure Condition:** If User selects "I don't remember" or answers incorrectly once.

5.2 Tier 2: The "Semantic" Fallback (Medium)

- **Trigger:** Activated immediately upon Tier 1 failure.
- **Logic:** Shift from *Specifics* to *Categories/Locations*.
- **Prompt:** *"Okay, let's try a broader question. You made a purchase in **Malibu, CA** last weekend. Was it related to: A) Groceries, B) Entertainment, C) Automotive?"*
- **Security:** High (Requires knowing location history).
- **Success:** Unlocks the account but flags for "Soft Monitoring" (transactions > $1,000 require re-verification).

5.3 Tier 3: The "Degraded" State (Fail)

- **Trigger:** Failure of Tier 2.
- **Logic:** The user is likely an imposter or has significant memory lapses.
- **Action:**
    1. Lock the specific transaction.
    2. Trigger standard "Step-Up" (SMS/Email + Call Center PIN).
    3. **Adversarial Logging:** Record the session parameters to train the "Suspicion Model."

Evaluation & Metrics

We will not just measure "Accuracy." We will measure "Authentication Quality."

Metric 1: Grounding Score (Hallucination Check)

- ***Definition:*** Does the generated question *actually* match the database record?
- ***Method:*** An independent "Judge LLM" compares the generated question against the SQL row.
- ***Pass Criteria:*** 100% match. (If the user went to Starbucks, the Agent must not ask if they went to Peet's).

Metric 2: Distractor Plausibility (Difficulty)

- ***Definition:*** How hard is it to guess the answer randomly?
- ***Method:*** We will compute the Cosine Similarity between the correct answer and the distractors.
    - ***Bad:*** Correct="Starbucks", Distractor="Home Depot" (Similarity: 0.2 - Too easy).
    - ***Good:*** Correct="Starbucks", Distractor="Dunkin" (Similarity: 0.8 - Good challenge).

Metric 3: User Friction (Simulated)

- ***Definition:*** Average time to answer.
- ***Goal:*** < 10 seconds. (Text-based questions are faster than checking an email for a code).

Weekly Execution Plan

Phase 1: Initiation & Architecture

- **Team & Charter**
    - **Action:** Create GitHub Repo. Define the role: "Security Product Analyst."
    - **Deliverable:** Update README.md with the "Dynamic Verification" problem statement.
- **AI-Native Cloud (Snowflake)**
    - **Action:** Load your synthetic dataset (PaySim or custom CSV) into Snowflake.
    - **Action:** Set up the connection between Python (local or Colab) and Snowflake.
- **Data Strategy & Governance**
    - **Action:** Define "PII masking." Even though you are the bank, your LLM prompt should probably mask the exact account number before sending it to OpenAI/LLM provider.
    - **Deliverable:** A governance doc explaining how you prevent the LLM from leaking transaction data in its output.

Phase 2: Core Development

- **Reproducibility**
    - **Action:** Create a standard Python script that ingests a CSV and outputs a "Transaction Summary" string.
- **Trust & Benchmarking**
    - **Action:** Define Success.
        - *Metric 1:* **Correctness** (Does the question match the data?).
        - *Metric 2:* **Distractor Quality** (Are the fake answers realistic?).
- **UX & HCD**
    - **Action:** Design the "fallback." If the user forgets the answer, what happens? (Maybe the Agent offers a different question).
- **Multimodal (Optional)**
    - *Skip or Adapt:* Maybe the challenge involves an image? "Select the logo of the coffee shop you visited." (Wanna keep it text-based if this is too hard).
- **Temporal Intelligence**
    - **Crucial Week:** Focus on the *Time* aspect. Ensure the LLM understands "Last Tuesday" vs "Two weeks ago."
- **Knowledge Graph (The Core Tech)**
    - **Action:** Implement **GraphRAG**. Use LlamaIndex to build an index over the transaction history. This allows the LLM to traverse data accurately ("What did I buy *after* I went to the gas station?").

Phase 3: Integration & Launch (Weeks 10-16)

- **Pipeline Integration**
    - **Action:** Connect Snowflake Data $\to$ Graph Index $\to$ LLM $\to$ Streamlit UI.
- **Research-A-Thon**
    - **Action:** Create a poster showing a "Hacker" trying to guess the answer vs. the "Owner" easily answering.
- **Advanced RAG**
    - **Action:** Fine-tune the prompt to ensure the "Voice" of the agent is professional and secure (Banks shouldn't sound like casual chatbots).

- **Ethics & Fairness**
    - **Action:** Test for bias. Does the system generate harder questions for certain merchant categories?
- **Final Submission**
    - **Deliverable:** Working Streamlit Demo + GitHub Repo.

References

https://www.proof.com/blog/the-future-of-digital-identity-why-mdocs-verifiable-credentials-and-dids-matter

https://www.huntress.com/blog/biggest-data-breaches

https://datadome.co/guides/account-takeover/what-are-fullz-how-do-fullz-work/#:~:text=The%20price%20of%20fullz%20sets,to%20pay%20for%20fullz%20sets.

https://auth0.com/blog/why-sms-multi-factor-still-matters/

https://www.cunastrategicservices.com/content/why-knowledge-based-authentication-solutions-are-failing-and-whats-next

https://www.rippling.com/blog/security-questions

https://arxiv.org/html/2511.08505v1#:~:text=Among%20structure%2Dbased%20methods%2C%20edge2024local,information%20essential%20for%20answering%20queries.