Self-Evolving Systems: Moving Beyond Deterministic Interfaces to Adaptive Generative Interfaces

Piyush Arora Ram Vivekananda Wendy Yun

Colby Hawker Jaime Sonoda

Google

{piyusharora, vivekanandaram, wendyyun,

colbyhawker, jsonoda}@google.com

Executive Summary

The field of Human-Computer Interaction is approaching a critical inflection point, moving beyond the era of static, deterministic systems into a new age of self-evolving systems. We introduce the concept of **Adaptive generative interfaces** that move beyond static artifacts to autonomously expand their own feature sets at runtime. Rather than relying on fixed layouts, these systems utilize generative methods to morph and grow in real-time based on a user's immediate intent.

The system operates through three core mechanisms: **Directed synthesis** (generating new features from direct commands), **Inferred synthesis** (generating new features for unmet needs via inferred commands), and **Real-time adaptation** (dynamically restructuring the interface's visual and functional properties at runtime). To empirically validate this paradigm, we executed a within-subject (repeated measures) comparative study (N=72) utilizing 'Penny,' a digital banking prototype. The experimental design employed a counterbalanced Latin Square approach to mitigate order effects, such as learning bias and fatigue, while comparing Deterministic interfaces baseline against an Adaptive generative interfaces. Participant performance was verified through objective screen-capture evidence, with perceived usability quantified using the industry-standard System Usability Scale (SUS).

The results demonstrated a profound shift in user experience: the Adaptive generative Interface achieved a **System Usability Scale (SUS)** score of **84.38** ('Excellent'), significantly outperforming the Deterministic version's score of **53.96** ('Poor'). With a statistically significant mean difference of **30.42 points** ($p < 0.0001$) and a large effect size (d=1.04), these findings confirm that reducing 'navigation tax' through adaptive generative interfaces directly correlates with a substantial increase in perceived usability.

We conclude that deterministic interfaces are no longer sufficient to manage the complexity of modern workflows. The future of software lies not in a fixed set of pre-shipped features, but in dynamic capability sets that grow, adapt, and restructure themselves in real-time to meet the specific intent of the user. This paradigm shift necessitates a fundamental transformation in product development, requiring designers to transcend traditional, linear workflows and evolve into 'System Builders'—architects of the design principles and rules that facilitate this new age of self-evolving software.

1 Introduction

The field of Human-Computer Interaction is approaching a critical inflection point. Traditional interfaces with predefined sets of experiences that they enable with limited customizability, function as static artifacts. We are moving beyond this era of deterministic software into a new age of "AI-morphic" software, where the application ceases to be a fixed construct and instead functions as a **self-evolving system**. This transition represents a massive opportunity to redefine the next generation of user experiences, moving away from software that is merely built & shipped, and toward software that is grown and adapted in real-time to the end user needs.

Our research seeks to define and explore this critical new frontier. This paper answers the question: "How will AI-morphic generative interfaces fundamentally redefine human-computer interaction?". We present a comprehensive analysis of the core concept of adaptive generative interfaces. We argue that software must evolve from a fixed product into an adaptive system that responds to user needs in real-time.

To assess the potential of this paradigm, we conducted a comparative study using "Penny," a hypothetical banking application prototype designed to resolve the limitations of deterministic software. In this model, the application utilizes adaptive generative interface to morph and evolve based on immediate user context rather than relying on fixed layouts. The study examined the efficacy of this concept, where the software autonomously expands its own feature set—such as constructing entirely new, functional UI for spending limits in response to user intent. Furthermore, we tested real-time adaptability through the instant generation of high-contrast accessibility themes that were never explicitly pre-built by the engineering team. By analyzing key metrics such as perceived usability, ease of use of the product, task efficiency, and user trust, this paper demonstrates how generative interfaces enable an application to grow with its user, offering a level of personalization and responsiveness that deterministic interfaces cannot offer.

1.1 Adaptive Generative Interfaces Prototype

To evaluate this paradigm, we developed **"Penny,"** a banking application utilized in two distinct configurations for a comparative study:

- **Deterministic version:** Uses a traditional interface where all features, navigation paths, and visual styles are predefined. Functionality is limited strictly to what was hard-coded during development, representing the "fixed" nature of current software.
- **Adaptive generative version:** This version employs an adaptive generative interface where the UI evolves its functional and visual properties at runtime The system responds to user intent through three core mechanisms: **Directed Synthesis**, **Inferred Synthesis**, and **Real-Time Adaptation**.

2 System Architecture

To support the capabilities of the adaptive generative interface—where UI components are synthesized rather than merely retrieved, we required an architecture that fundamentally differs from the traditional Model-View-Controller (MVC) approach. In the Adaptive Generative version, the frontend functions as a dynamic canvas controlled by a probabilistic inference engine.

We propose a **Hybrid generative architecture** that acts as an orchestration layer between the user's intent and the application's underlying logic. This layer determines the necessity and form of interface modifications by **identifying functional gaps within the current user context.** This evaluation ensures that generative synthesis is only triggered when a functional gap is detected, maintaining system stability while resolving user friction. This system is designed to intercept natural language inputs, determine the necessary interface modifications, and execute

them in real-time without destabilizing the core application state or security protocols.

The architecture consists of three primary layers: Intent Recognition, Component Orchestration, and Runtime Rendering. The interaction between these layers is visualized below.

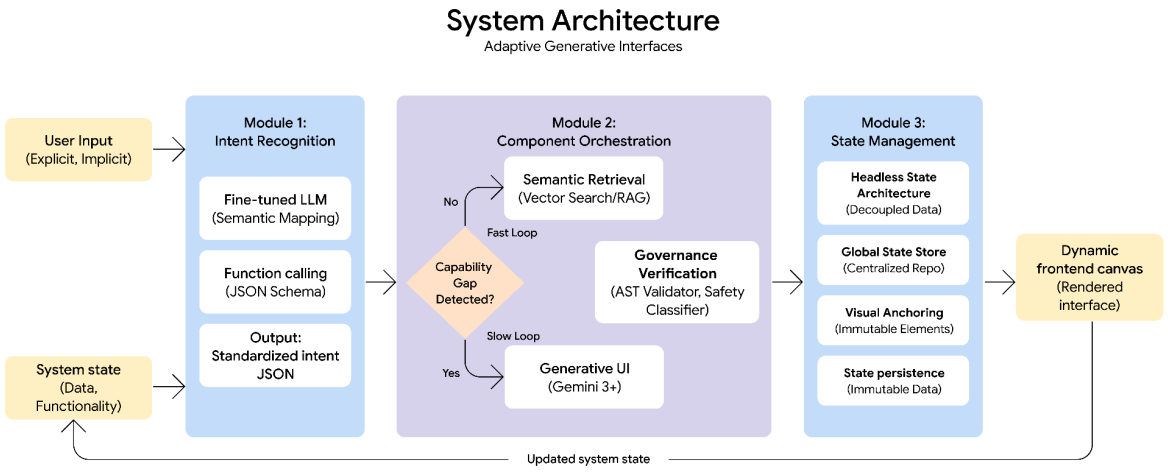

Figure 1: System Architecture

Module 1: Intent Recognition: The system maps model outputs to UI requirements (e.g., intent classification of the response).

- The following signals are derived:
  - Explicit input: direct natural language commands and determinate interactions
  - Implicit input: the user's natural language prompts
  - System state: the current functional status of the application and state of data (eg. account balances)
- The following technologies are used:
  - Fine-tuned Large Language Model (LLM) to perform the semantic mapping or intent classification
  - Function calling to convert heterogeneous inputs into a standardized JSON schema that defines the function of the required interface, E.g.:
    - { urgency: "high", task: "verification" })

Module 2: Component Orchestration: The logic for selecting or generating the "best fit" UI widget (tables, graphs, interactive modules).

- The following strategies are executed:
  - **Semantic retrieval of existing UI elements (Fast loop):** Match the intent against a repository of pre-validated UI elements.
  - **Generative UI elements (Slow loop):** Construction of novel UI widgets when a "capability gap" is detected between user intent and existing assets. This construction is performed via a specialized code generation model.
  - **Governance Verification:** Strict validation of code logic to ensure safety before the interface is rendered to the user

- The following technologies are used:
    - **Vector Embeddings & Search**: Evaluate the semantic distance (similarity) between the user's intent and the metadata of existing components; In some cases, this may employ a RAG (Retrieval Augmented Generation) architecture.
    - **Code Generation Model:** A high-fidelity reasoning model (specifically the Gemini 3 class) is required to synthesize executable front-end code. This selection is critical; while earlier-generation models suffered from UI rendering output errors as high as 60%, the latest reasoning tier (Gemini 3) has reduced this error rate to 0% **(Leviathan et al., 2025)**. This jump in reliability represents the "emergent capability" that makes the Slow Loop (generative synthesis) technically feasible for safe runtime production.
    - **Abstract Syntax Tree (AST) Validator**: Parse generated code for syntax errors and block unauthorized API calls or security vulnerabilities.
    - **Safety classification model**: A specialized LLM will be used to check the inputs for malicious attack (e.g., prompt injection) or unexpected unsafe outputs.

Module 3: State Management: the system maintains consistency so the user doesn't feel disoriented when the layout changes.

- The following mechanisms are used:
    - Visual Anchoring: The preservation of specific, immutable elements (e.g., global navigation, headers) to provide grounding during layout shifts)
    - State Persistence: Ensuring that transactional data remains immutable regardless of how the presentation layer evolves.
- The following technologies are used:
    - Headless State Architecture: The total decoupling of the data layer from the presentation layer (the user view) to prevent data loss during UI morphing.
    - Global State Store: A centralized data repository that facilitates the instant propagation of data across a shared dynamic view

3 Prototype

To validate the paradigm of self-evolving systems, we developed "Penny," a high-fidelity functional prototype of a digital banking application. Banking was selected as the domain for this study due to its reliance on complex, multi-dimensional data and the necessity for high trust and security.

We built 2 versions of the prototype: **Deterministic Version** and **Adaptive Generative Version**. Both versions utilized the same underlying user profile: a customer with a tenure holding checking, savings, and credit card accounts.

3.1 Scope and Tooling

The 'Penny' prototype was built as a fully functional application rather than a static mockup. To enable the Adaptive generative version, we integrated a large language model (LLM) backend that allows the system to write and render new UI components in real-time based on user context.

3.2 Deterministic Version

In the **Deterministic Version**, the interface functioned as a static artifact. Users interacted with the application through standard, pre-defined navigation paths. We observed specific friction points inherent to this model:

- **Cognitive Load in Data Comparison:** To compare monthly spending between 2025 and 2024, users were required to manually toggle a timestamp dropdown. This forced the user to switch back and forth (context switching) to mentally compare data points, rather than viewing them simultaneously.

- **Navigation Friction:** Setting a subscription limit required a multi-step navigation sequence: users had to locate the “Budget/Limit” on settings page, find the specific category, and manually input data.
- **Accessibility Gaps:** When users sought a high-contrast viewing mode, they were limited to the pre-shipped "Light" and "Dark" themes. Because the product and engineering team had not explicitly shipped a high-contrast theme, the user’s need went unmet.

3.3 Adaptive Generative Version

In the Adaptive generative version, the application autonomously evolves based on user intent. This was facilitated through an "Ask AI" entry point, allowing for natural language command and control. The prototype demonstrated three core generative capabilities:

3.3.1 On-Demand Data Visualization

Unlike the static charts in the control version, the generative interface allowed users to manipulate data presentation via natural language. When prompted with, *"Can you update this chart to show the monthly spending for 2025 compared to 2024,"* the system recognized the implicit intent for comparison. Instead of forcing a manual toggle similar to the deterministic version, the system generated a comparative bar chart, plotting both datasets on a single graph to facilitate immediate insight.

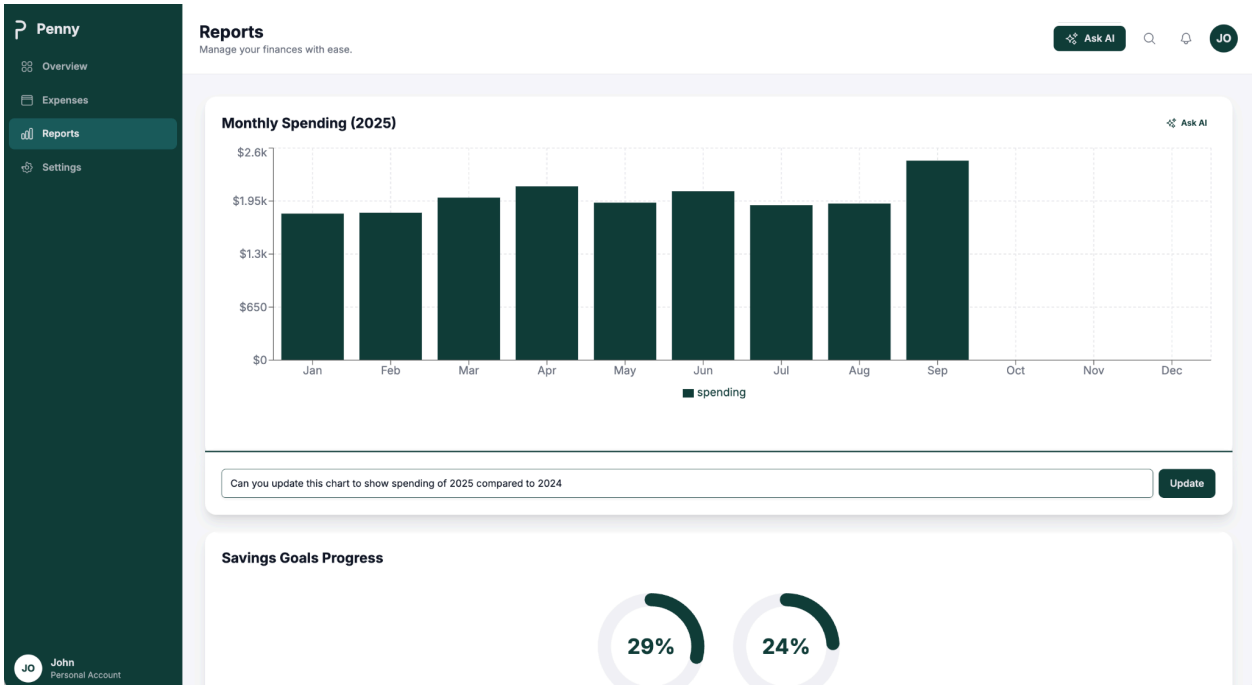

Figure 2: User prompt input for on-demand visualization

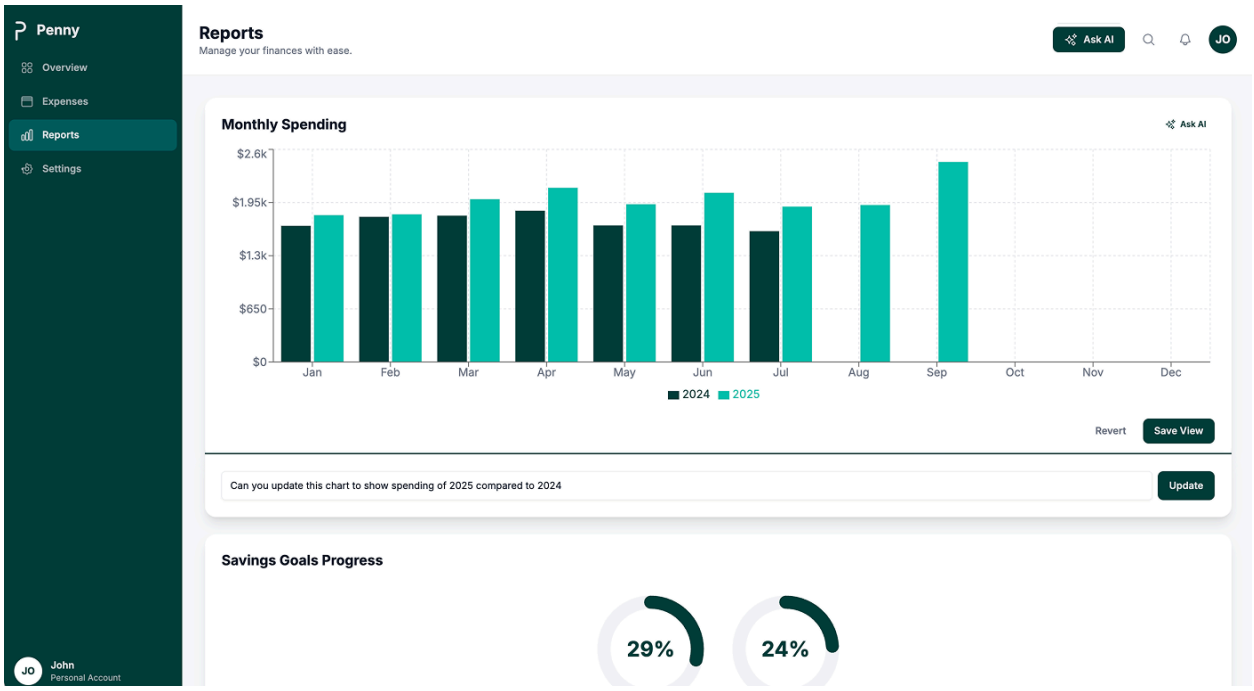

Figure 3: Updated UI with comparative bar chart for spending in 2025 vs 2024

3.3.2 Autonomous Feature Expansion

We tested the system's ability to bridge "capability gaps" by autonomously expanding its own feature set at runtime. This was observed across two primary contexts:

- **Directed Synthesis:** When users provided a direct command (*"Please set my subscription spending limit... and add a custom widget to track"*), the system executed the requisite backend logic while simultaneously constructing a novel tracking component on the dashboard. This demonstrated the system's capacity to translate explicit natural language commands into functional UI components.

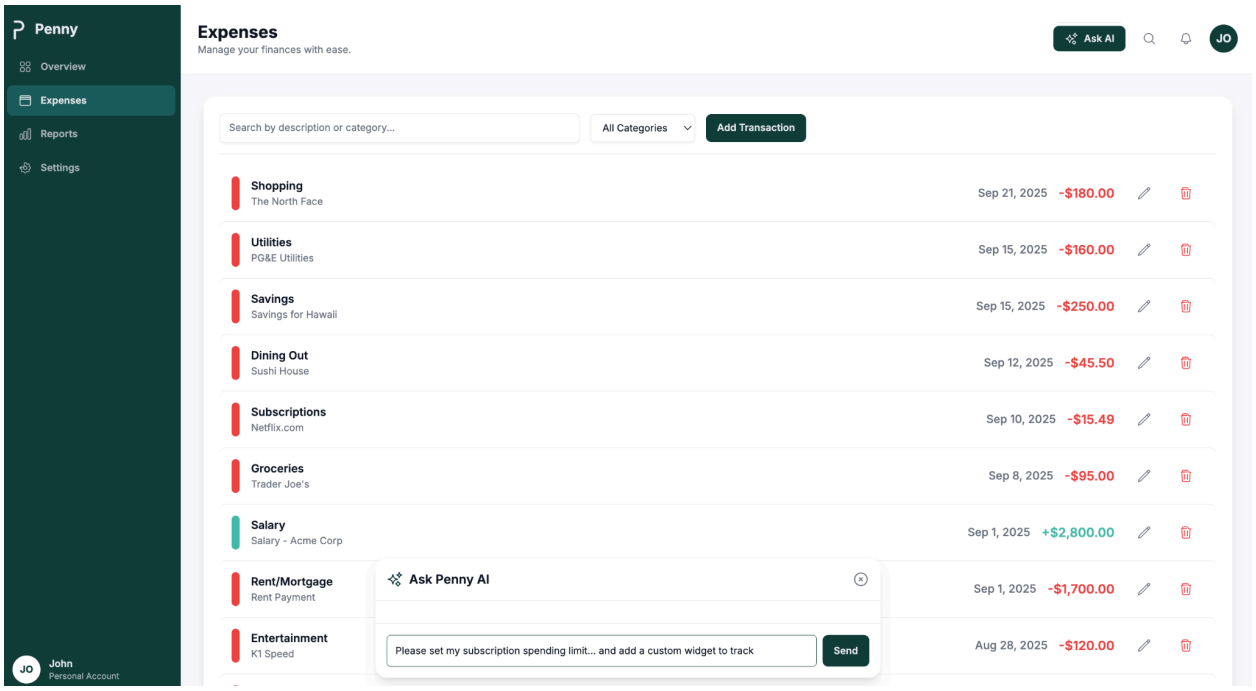

Figure 4: User request to set a spending limit and tracking widget.

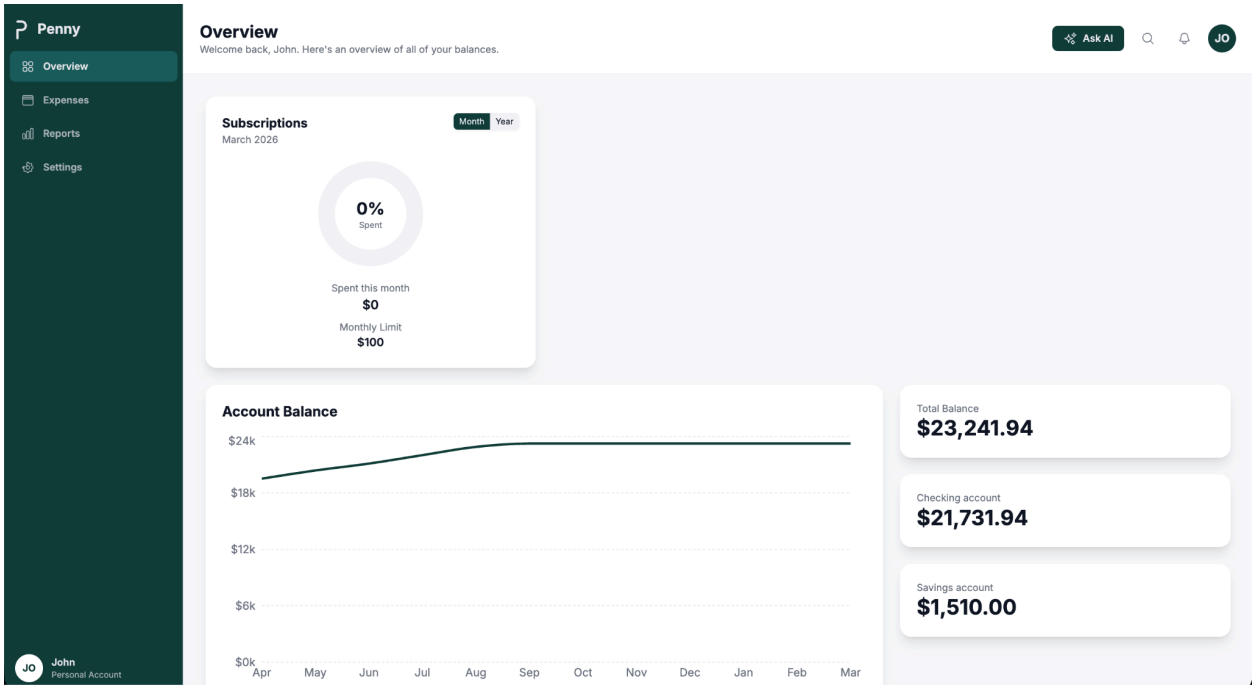

Figure 5: Dashboard updated with a new subscription tracking module.

- **Inferred Synthesis:** To evaluate the system's inferential capabilities, users offered high-level, goal-oriented prompts (*"I'm trying to be better with my finances"*). Although no specific interface was requested, the system inferred the need for structured support and autonomously constructed a "Smart Tips" module. This generated interface synthesized spending insights and embedded context-aware Call-to-Actions (CTAs), effectively building a custom financial management tool based entirely on implicit intent.

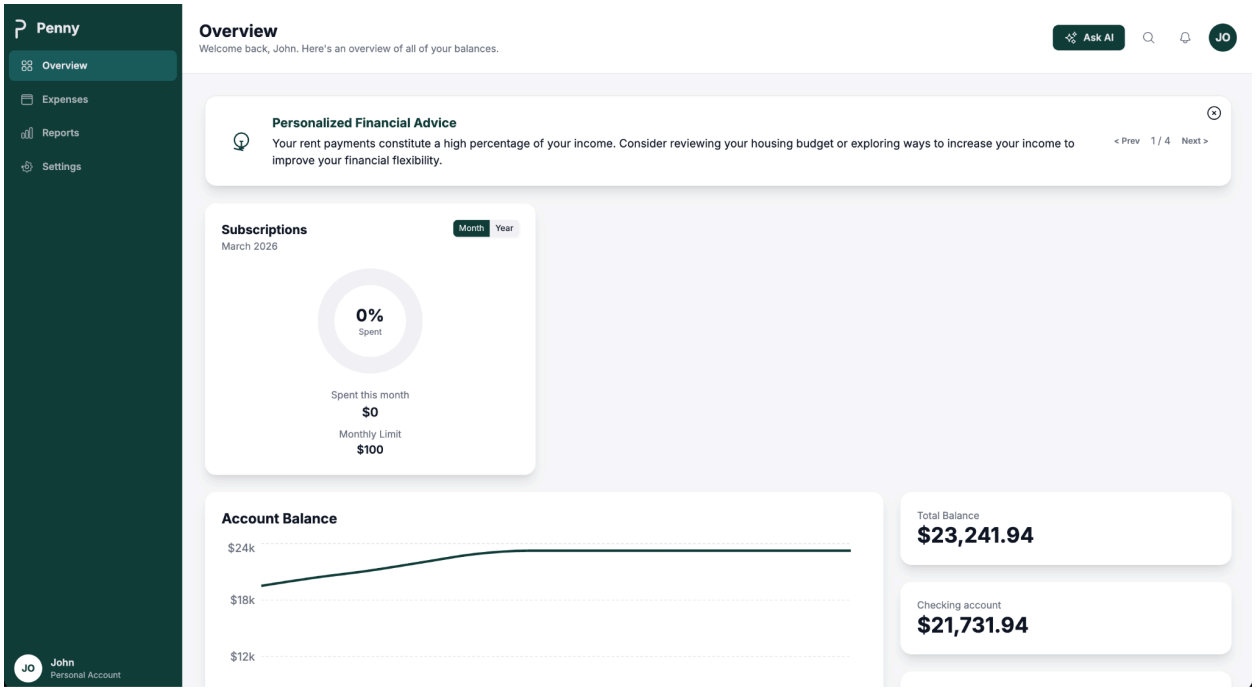

Figure 6: Autonomously generated "Smart Tips"

3.3.3 Real-Time Adaptation

The prototype demonstrated the ability to dynamically alter its structural and stylistic properties to meet immediate user needs.

- **Adaptive Layout Restructuring:** When the user prompted, *"I'd like to be able to collapse the navigation, so I have more space to view the charts,"* the system modified the global layout to add a collapsible side menu. This "hamburger" menu was not a pre-built feature of the application. Instead, the system generated the code for the menu on the fly, allowing the user to customize their workspace layout to meet their immediate needs.
- **On-Demand Visual Adaptation:** When prompted with *"I didn't bring my glasses today. Can you create a high contrast theme, so I can read better?"*, the system did not retrieve a premade theme, which didn't exist in this case. Instead, it dynamically adjusted the CSS variables of the application in real-time to generate a high-contrast theme that had never been explicitly developed by the product and engineering team.

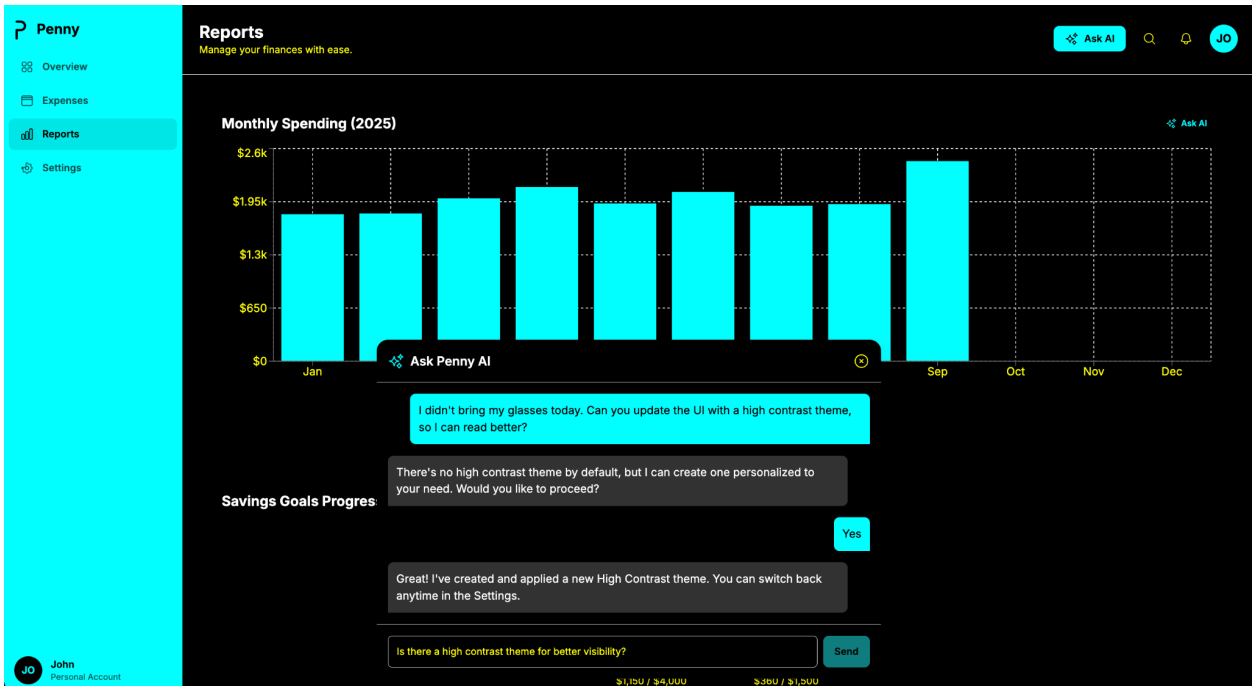

Figure 7: Real-time generation of high-contrast accessibility theme.

3.4 Design Rationale

The prioritization of these specific modules—On-Demand Data Visualization, Autonomous Feature Expansion, and Real-Time Adaptation was intentional. They represent the three distinct layers of UI generation: **Content Adaptation** (charts), **Functional Expansion** (widgets), and **System-wide Adaptation** (themes). By prototyping these layers, we were able to measure the impact of generativity on perceived usability and overall user satisfaction.

4 Methodology

4.1 Experimental Design

This study employed a **within-subject (repeated measures) experimental design**. Each participant interacted with [Deterministic version] and [Adaptive generative version] to allow for a direct comparison of usability metrics while controlling for individual differences in baseline technical proficiency.

To mitigate **order effects** (learning bias and fatigue), the presentation of the systems was **counterbalanced** using a Latin Square approach. Participants were randomly assigned to one of two groups:

- **Group 1:** Task completion on Deterministic version followed by Adaptive generative version.
- **Group 2:** Task completion on Adaptive generative version followed by Deterministic version.

4.2 Participants

A total of 72 participants were successfully recruited and distributed across the two experimental conditions (Group 1: 35; Group 2: 37). Inclusion criteria required participants to use laptops to complete the study, and to be frequent users of digital banking applications at least once every two days. They also had to be familiar with day-to-day online banking tasks, such as checking balances, viewing transaction history, and managing budgets, and must use a

mix of foundational products (Checking, Savings, and Credit Card).

4.3 Materials and Apparatus

The entire experimental workflow, from screening to final data collection, was managed through an integrated survey system. The study was facilitated remotely, using this environment to deliver standardized task instructions and aggregate participant responses. To supplement these metrics, screenshot data was collected for each task the participant completed.

4.4 Testing Procedure and Task Execution

The experimental session required participants to interact with both the Adaptive generative version and the Deterministic version. The task load was structured as follows:

- Deterministic version: Participants were assigned 8 specific tasks.
- Adaptive generative version: Participants were assigned 9 specific tasks.

For each design, a validation-based approach was employed. Participants were required to execute each task and upload a screen capture of the final state as objective evidence of task completion. This ensured data integrity and confirmed that participants engaged with the core functional components of each interface.

Upon the successful completion of all tasks within a specific design, participants were immediately administered the System Usability Scale (SUS) to capture their perceived usability, followed by a single-item Overall Satisfaction question. For each of the tasks participants were also asked to provide qualitative feedback.

4.5 Inclusion Criteria and Data Cleaning

To maintain the rigor of the within-group comparison, a strict completion requirement was enforced. Only data from participants who successfully completed the full suite of tasks for both the Adaptive generative and Deterministic version were included in the final analysis. Partial completions were excluded to ensure a balanced, pairwise comparison of the two systems.

4.6 Measures: System Usability Scale (SUS)

The primary quantitative metric for perceived usability was the **System Usability Scale (SUS)**, a validated, 10-item Likert scale.

- **Administration:** The SUS was administered immediately following the completion of tasks for each system to capture fresh impressions.
- **Scoring:** In accordance with Brooke (1986), individual items were scored from 0 to 4. For odd-numbered items, the score is the scale position minus 1. For even-numbered items, the score is 5 minus the scale position. The sum of these scores was multiplied by 2.5 to produce a final score ranging from 0 to 100 **(Brooke, 1986)**.
- **Interpretation:** Scores were evaluated against the industry-standard benchmark: a score above **68** is considered "average," while scores exceeding **80** are categorized as "excellent" (Grade A).

4.7 Data Analysis

The descriptive statistics for the System Usability Scale (SUS) scores across both design conditions are summarized in Table 1.

Table 1: Descriptive statistics

Metric	Deterministic version	Adaptive generative version
Mean SUS Score	53.96	84.38
Standard Deviation (SD)	26.47	17.76
Standard Error (SE)	3.12	2.09
Median SUS Score	52.5	90
95% Confidence Interval	[47.74, 60.18]	[80.20, 88.55]

According to SUS industry benchmarks **(Bangor et al., 2009)**, the SUS scores and their interpretation is as follows

- Deterministic version which has a SUS score of (53.96) is classified as "Poor/Marginal" (Grade F).
- Adaptive generative version which has a SUS score of 84.38 is classified as "Excellent" (Grade A) **(Bangor et al., 2009).**

4.7.1 Data Interpretation and Variability

Analysis of the central tendencies reveals that the **Adaptive Generative Version** achieved a substantially higher mean score (M = 84.38) compared to the **Deterministic Version** (M = 53.96). The median for the Adaptive generative version (90.00) suggests a negatively skewed distribution, indicating that a majority of participants reported very high usability satisfaction.

In contrast, the Deterministic version exhibited a higher degree of variability (SD = 26.47) than the Adaptive generative version (SD = 17.76). This greater dispersion in the Deterministic version suggests that user experiences were inconsistent, with some participants encountering significantly more friction than others. The non-overlapping **95% Confidence intervals** further provide a strong visual and mathematical indication that the performance gap between the two prototypes is robust and unlikely to be the result of sampling error.

4.7.2 Analysis of Statistical Assumptions

To determine the appropriateness of parametric testing, the normality of the distribution of differences between the two conditions was evaluated. A **Shapiro-Wilk test** was performed on the paired differences, yielding $W = 0.978$, $p = 0.247$. Since $p > 0.05$, the assumption of normality for the paired differences is met, justifying the use of a parametric **Paired samples T-test**.

4.7.3 Inferential Statistics and Effect Size

A two-tailed Paired-Samples T-test was conducted to examine the statistical significance of the usability improvements observed in the Adaptive generative version compared to the Deterministic version. The analysis revealed a highly significant difference in mean SUS scores

- **T-statistic:** $t(71) = 8.806$
- **Degrees of Freedom** (df)**:** 71
- **P-value:** $p < 0.0001$ (5.36×10^{-13})
- **Mean Difference:** 30.42 points
- **Effect Size (**Cohen's d**):** 1.04

A Cohen's *d* of 1.04 represents a **"Large" effect size** (typically > 0.80) **(Cohen, 1988)**. This indicates that the improvement in usability with the Adaptive generative version is not only statistically significant but practically substantial.

4.8 Summary

A paired-samples t-test was conducted to compare the usability of the Deterministic and Adaptive generative versions using System Usability Scale (SUS) scores. Results revealed a statistically significant difference in favor of the Adaptive generative version ($M = 84.38$, $SD = 17.76$) compared to the Deterministic version ($M = 53.96$, $SD = 26.47$); $t(71) = 8.806$, $p < .001$. The calculated effect size (Cohen's $d = 1.04$) indicates a large magnitude of effect. These findings suggest that the Adaptive generative version significantly enhances the user experience, moving the system from a 'Poor' usability grade to an 'Excellent' grade.

4.8.1 Non-Parametric Verification

To be extra rigorous, we also ran a **Wilcoxon Signed-Rank test** (which does not assume normality). The results remained highly significant: $Z = 140.5$, $p < .001$ (Wilcoxon, 1945). This confirms the findings are robust regardless of the statistical method used.

5 Discussion

The results of this study signal a potential paradigm shift in Human-Computer Interaction. The transition from a deterministic interface to an adaptive generative interface resulted in a **30-point increase** in **System Usability Scale (SUS)** scores, moving the application from a **"Poor" rating (53.96)** to **"Excellent" (84.38)**.

This massive effect size ($d = 1.04$) validates our core hypothesis: we are moving beyond the era of static artifacts into the age of **self-evolving systems**. The data suggests that the friction users experience today is often a failure of rigidity.

While traditional deterministic systems force users to learn the system's mental model, the adaptive generative systems invert this dynamic, forcing the system to learn the user's mental model.

5.1 The Mechanism of Usability: Reducing Friction through Agency

The superior performance of the Adaptive generative version can be attributed to the fundamental decoupling of **capability** from **navigation**. In the Deterministic version, the user bore the "cognitive tax" of mapping their intent to the system's static hierarchy (Sweller, 1988) —forcing them to memorize interaction paths to execute simple tasks like comparing dates or setting limits.

In the Adaptive generative version, this dynamic was inverted. By utilizing On-demand data visualization, Autonomous feature expansion and Real-Time adaptation mechanisms, the system did not merely "find" a feature; it **synthesized the necessary interface elements in real-time** directly around the user's intent. This confirms that the usability gains were driven by a shift in architectural agency: the system evolved from a static map that the user must traverse into a responsive environment that configures itself to the user's immediate goal.

5.2 Implications: The Era of Self-evolving Systems

For decades, software has been "built and shipped"—a fixed construct delivered to millions of users. Our findings challenge the viability of this approach. The success of the Real-time adaptation (creating a high-contrast theme on the fly) demonstrates that software can now be "grown" at runtime.

The future of software architecture lies in self-evolving systems. In this model, the application ceases to be a static

product and instead becomes an adaptive system that continuously reshapes itself around the specific needs of the user.

5.3 Technological Viability and "Emergent Capabilities"

While this study validates the usability of adaptive generative interfaces, the feasibility of deploying them at scale is inextricably linked to model performance. As noted in the Google GenUI study (Leviathan et al., 2025) , robust UI generation is an "emergent capability" that only stabilizes with the most recent class of reasoning models.

The data indicates that while smaller, earlier-generation models suffered from UI rendering output errors as high as 60%, the latest generation (Gemini 3) reduced this error rate to 0% and an Elo score of 1706.7 (Leviathan et al., 2025). This suggests that while the underlying generative models have reached the necessary threshold of reliability, the viability of self-evolving software now depends on the robustness of the surrounding architecture to manage state, security, and scale.

5.4 Future Outlook

This study validates the immediate usability benefits of Adaptive generative interfaces. However, future research must move beyond single-session usability to examine longitudinal effects. As users become accustomed to software that builds itself, will their trust increase, or will the lack of permanence lead to anxiety? Furthermore, solving the technical latency of real-time generation will be the deciding factor in moving this architecture from research prototypes to production reality.

6. Challenges

The rise of generative AI has ushered in the exciting possibility of experiences that can morph themselves in real time to cater to the user needs. While this technology holds the promise of truly personalized and intuitive user experiences, it also presents a host of significant challenges that must be addressed. These challenges span across usability, technical implementation, and ethical considerations.

6.1 User Experience and Usability

At the forefront of these challenges are concerns surrounding the overall UX of adaptive generative interfaces.

- **Loss of user control and predictability:** An interface that constantly changes can be disorienting. Users rely on spatial memory and consistent layouts to navigate applications efficiently. If buttons and menus move or change unexpectedly, it can lead to a steep learning curve and a sense of being out of control. Striking a balance between adaptation and predictability is crucial.
- **Cognitive overload:** While the goal of an adaptive generative interface is to simplify the user's journey, a poorly designed one can have the opposite effect. Presenting too many on-the-fly options or altering the existing interfaces too frequently can overwhelm users and increase cognitive load.
- **Consistency and cohesion:** Generative models historically have tendencies to produce inconsistent UI elements, resulting in a disjointed user experience. Maintaining a consistent design language and ensuring all interfaces adhere to established usability heuristics is a significant hurdle.

6.2 Technical and Implementation Hurdles

The technical complexities of building and deploying self-evolving systems are substantial.

- **Performance and Latency:** Generating UI elements on the fly requires significant computational resources, which can lead to performance issues and slow loading times. Optimizing generative models to ensure real-time responsiveness is a major technical challenge.
- **Testing and Quality Assurance:** Traditional methods of software testing are ill-suited for the unpredictable nature of adaptive generative interfaces. It is difficult to create test cases that cover all

possible UI configurations, necessitating new testing methodologies to ensure reliability.
- **Data Quality and Bias:** The output of generative AI is dependent on its training data. If the data is biased or incomplete, the resulting UI may perpetuate those biases, leading to an unfair or exclusionary user experience.

6.3 Ethical and Trust-related Concerns

The ability of AI to personalize interfaces on the fly raises important ethical questions.

- **Transparency and Explainability:** Users may be wary of an interface that changes without explanation. It is critical for systems to be transparent about why changes are being made to maintain user trust.
- **Privacy:** To personalize an interface, the AI needs to collect and analyze user data. Users need to be informed about data usage and possess the ability to opt-out or control the level of personalization.

7. Role of Designer in Age of AI

The role of the UX designer is undergoing a significant transformation, moving from a traditional, linear workflow to a more dynamic, AI-integrated partnership. In the traditional UX role, the designer's process was conventional: design concepts were created, followed by a detailed workflow and then execution, with little change to the fundamental role. However, the rise of Generative AI has begun to blur the lines between product management, engineering, and UX.

With "vibe coding" tools that translate natural language prompts into functional prototypes, anyone can now build live prototypes before a formal product requirements document exists. This shifts the process from a rigid "Design > Spec > Execute" model to an agile "Context > Live Prototype > Execute" cycle.

Consistent with the "Generative UI" framework proposed by Google Research (Leviathan et al., 2025), which demonstrates how "carefully crafted system instructions" enable AI to function as an on-demand product team, we argue that the definition of design is shifting from "crafting static artifacts" to "defining system constraints" and declaring intent. This shift marks the emergence of the "SuperIC" (Product Designer + PM + Visual Designer + UX Engineer). This new breed of designer acts as a **System Builder**, expanding the designer’s scope from designing static layouts to architecting the fundamental behaviors and boundaries of a self-evolving system."

8. Conclusion

This research set out to answer a foundational question: "How will AI-morphic generative interfaces fundamentally redefine human-computer interaction?". To explore this, we moved beyond theoretical frameworks and conducted a rigorous empirical study using "Penny," a functional banking prototype designed to test the efficacy of **self-evolving systems**.

Our findings validate that the era of deterministic, "one-size-fits-all" software is reaching its limit. The significant performance gap observed between the deterministic and adaptive generative versions confirms that when software is capable of **autonomously expanding its feature set in real-time** —it can effectively eliminate the friction that burdens traditional user experiences.

The implications of this shift are profound. We have demonstrated that the definition of a "finished product" is changing. Software is no longer a static artifact built once and shipped to all; it is a system that grows, adapts, and restructures itself to meet the specific intent of the user.

Realizing this future requires addressing the significant hurdles that remain. While challenges in latency, predictability, and governance must be solved , the path forward is clear: the next generation of best-in-class user experiences will not be designed *for* the user, but evolved *with* them.

References

Aaron Bangor, Philip T. Kortum, and James T. Miller. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123, 2009. URL https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/

John Brooke. SUS: A retrospective. *Journal of Usability Studies*, 8(2):29–40, 2013. URL https://uxpajournal.org/sus-a-retrospective/

Jacob Cohen. Statistical power analysis for the behavioral sciences. *Routledge*, 1988. URL https://www.taylorfrancis.com/books/9780203771587

Yaniv Leviathan, Dani Valevski, Matan Kalman, Danny Lumen, Eyal Segalis, Eyal Molad, Shlomi Pasternak, Vishnu Natchu, Valerie Nygaard, Srinivasan (Cheenu) Venkatachary, James Manyika, Yossi Matias. Generative UI and the transition to agentic systems: Emergent reasoning capabilities in frontend synthesis. *Google Research*, 2025. URL https://generativeui.github.io/static/pdfs/paper.pdf

John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988. URL https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1202_4

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. URL https://sci2s.ugr.es/keel/pdf/algorithm/articulo/wilcoxon1945.pdf