# ProtoAI: Model-Informed Prototyping for AI-Powered Interfaces (Extended Abstract)

**Hariharan Subramonyam**[1] , **Colleen Seifert**[2] , **Eytan Adar**[2]
[1]Stanford University
[2]University of Michigan
harihars@stanford.edu {seifert, eadar}@umich.edu

## Abstract

When prototyping AI experiences (AIX), interface designers seek effective ways to support end-user tasks through AI capabilities. However, AI poses challenges to design due to its dynamic behavior in response to training data, end-user data, and feedback. Designers must consider AI's uncertainties and offer adaptations such as explainability, error recovery, and automation vs. human task control. Unfortunately, current prototyping tools assume a black-box view of AI, forcing designers to work with separate tools to explore machine learning models, understand model performance, and align interface choices with model behavior. This introduces friction to rapid and iterative prototyping. We propose *Model-Informed Prototyping* (MIP), a workflow for AIX design that combines model exploration with UI prototyping tasks. Our system, *ProtoAI*, allows designers to directly incorporate model outputs into interface designs, evaluate design choices across different inputs, and iteratively revise designs by analyzing model breakdowns.

## 1 Introduction

When prototyping user interfaces (UI), designers work to transform end-user needs into interface specifications [Wood, 1997]. By taking a *top-down* approach, designers: (1) express user requirements as task-flows; (2) map task items into graphical user interface (GUI) objects; and (3) assess different task-to-GUI mappings against end-user needs to finalize the design [Wood, 1997]. For instance, to design a phone 'unlock' user experience (UX), the designer may consider interface alternatives—such as an alpha-numeric password, a numeric passcode, or pattern-based unlocking—to allow end-users to input identifying information for access. By assessing those alternatives against user needs (e.g., fast to unlock, secure, low cognitive effort to remember), the designer will finalize the UI design. However, when prototyping AI-powered applications, such a top-down approach is impractical [Yang *et al.*, 2020].

AI-powered applications bring additional challenges to UI prototyping. AI features introduce dynamic behavior due to the scope of training data, system use over time, and variations in input data individual users contribute and the potential to learn from outcomes. Thus, designers must identify the *interactions* between user task-flows and AI capabilities [Holmquist, 2017] in order to design the UI for AI experiences (AIX). By exploring AI's capabilities and limitations through prototyping, they need to design interface adaptations such as explanations for AI outputs, seamless handling of AI failures, and collecting user feedback to improve the AI [Amershi *et al.*, 2019]. In the process, AIX designers also need to assess interface choices against diverse users and contexts of use.

Unfortunately, current UI prototyping tools lack support for designing AI-powered interfaces. By assuming a 'black-box' view of AI, tools introduce *knowledge blindness* about necessary AI attributes during the design process [Subramonyam *et al.*, 2022]. Prototyping tools also lack support for iterative testing of AI features through a "fail fast, fail often [Yang *et al.*, 2019]" approach. For an *AI-powered* phone access using face identification (ID), current tools can at best show where to display the camera field of view on the interface and design static error messages. However, without exploring the AI's behavior first-hand, the designer may not know what inputs the AI needs (e.g., head frontal-view). They may fail to understand how accurately the AI can perform, when it might fail, and how to prompt users experiencing failure (e.g., by asking them to move closer to the camera). To prototype AI features, designers currently work with multiple tools to explore AI behavior, probe its capabilities and limitations, and evaluate their design with diverse user inputs (e.g., skin-tone, lighting conditions, camera angle, facial features such as beard, glasses, or a mask). This introduces friction to the rapid prototyping process [Beaudouin-Lafon and Mackay, 2009].

In this work, we propose *Model-Informed Prototyping* (MIP), a workflow that streamlines the AIX design process by combining model exploration and interface design tasks. In our system implementing MIP, *ProtoAI*, designers can directly run target machine learning models by providing in-

*The original paper was published in the 26th International Conference on Intelligent User Interfaces (IUI '21). The paper received the Best Paper Award. DOI: https://doi.org/10.1145/3397481.3450640

put data and then incorporate the model's outputs in their UI prototypes. Instead of placeholder content, ProtoAI's *design-by-instance* approach allows designers to experience the AI's behavior first-hand as they are designing. Further, ProtoAI automatically generates data previews of the UI for differing input data, allowing designers to evaluate designs for breakdowns across diverse scenarios and contexts of use. This enables them to decide how best to integrate AI features into end-user's tasks and offer necessary adaptations for AI's uncertainties. By extending the familiar design paradigm of current prototyping tools, ProtoAI allows designers to operationalize human-centered AI (HAI) guidelines within their created designs.

# 2 Related Work

A recommended workflow for UI prototyping consists of three phases: *design*, *test*, and *analysis*. A number of UI prototyping tools (including our own) follow this model [Klemmer *et al.*, 2000; Hartmann *et al.*, 2006]. Here, we describe requirements and techniques from prior literature for each phase as applied to AI-powered interfaces.

## 2.1 Design

Numerous guidelines exist to design AIX by considering human-centered needs and AI capabilities [Amershi *et al.*, 2019]. To *operationalize* them, designers need access to the AI model to map its characteristics to the UI syntax [Dellermann *et al.*, 2019]. For instance, in mixed-initiative design, AI systems automatically act on end-users' goals (when clear) and use interface 'dialog' to resolve any uncertainty [Horvitz, 1999]. However, the specific dialog in the UI depends on the underlying AI and input data-context. In this regard, prior work has looked at using data as a material for AI design [Helms *et al.*, 2018; Subramonyam *et al.*, 2021]. Just as engineers prototype ML models, designers can begin with 'minimum-viable-data' and iteratively incorporate additional data for diverse users and contexts [van Allen, 2018]. This allows prototyping of AI interfaces from the inside-out: from the data model to UI. In ProtoAI we allow designers to incorporate input data and ML model outputs into UI prototypes.

## 2.2 Test

AIX designers need to map AI-to-interface features, identify gulfs of execution and evaluation, and assess visual aesthetics for AI features. Further, they should evaluate whether their design is robust to AI's unpredictability [Holmquist, 2017]: How does the AI-infused interface react to a diverse set of data and contexts of use [van Allen, 2018]? Building on existing UX practice, designers may consider approaches such as constructing personas with varying quantitative data [Pruitt and Grudin, 2003]. Wizard of Oz (WoZ) testing is also effective for evaluating early-stage prototypes [Maulsby *et al.*, 1993; Browne, 2019], and a number of data-dependent systems implement digitally scaffolded 'wizards' for testing prototypes during design [Klemmer *et al.*, 2000; Hartmann *et al.*, 2006]. For instance, Suede implements electronically supported WoZ testing techniques that generate chat messages using test data [Klemmer *et al.*, 2000]. In ProtoAI we automatically generate interface alternatives by invoking built-in models with input data provided by designers. This lets designers experience the UI's design first hand.

## 2.3 Analyze

To analyze performance at the AI model level, engineers use summary statistics such as accuracy, precision, and recall. Tools exist for engineers to analyze the overall performance and look at individual data points to reason about model failures (e.g., [Amershi *et al.*, 2015]). Designers need similar analysis and visualization tools at the interface level that will allow them to identify mismatches in model behavior. For instance, D.tools offers a 'group analysis' mode aggregating data from multiple user sessions into one view [Hartmann *et al.*, 2006]. The What-if tool allows designers to see the confusion matrix for binary classifiers visually [Google, 2020]. Designers should also be able to incorporate subjective metrics at the intersection of model performance and UX. In ProtoAI we support subjective analysis through designer generated tags and visual summaries. During iterative prototyping, the goal is to identify breakdowns in design and offer fixes [Winograd *et al.*, 1986]. ProtoAI's instantiation of UI for different data points allows designers to analyze AI-feature breakdowns without performing mental simulations of differing data contexts.

# 3 Model-Informed Prototyping

As shown in Figure 1, ProtoAI's implementation of MIP consists of four main views: (1) an AI models and services view (these can be implemented AI services or models, or Wizard of Oz 'stubs'), (2) a data view to import diverse input data for model simulation, (3) a UI 'designer' view to visually construct the interface prototype, and (4) a data previews view to simulate the interface design across different input data contexts. To better understand how a designer might use ProtoAI to engage in MIP, let us follow Divya, an AIX designer who is prototyping a Face-ID-based phone unlock experience.

## 3.1 Set-Up

Divya opens the ProtoAI application in the web browser. The Models tab is open by default and shows all of the AI services and models that are available in the system (Figure 1a). Divya's company has already assigned an engineering team to the project, and they have been working on an initial version of the Face-ID model. Divya selects the company's Face-ID model and navigates to the Data tab. The Data tab will allow her to import input data for different personas and scenarios of use. As shown in Figure 1b, the Data tab consists of a main editable spreadsheet view and sidebar view for model configuration. The spreadsheet can consist of *input data columns*, *feature/parameter columns*, *AI output columns*, and *derived (calculated) columns*. The sidebar view shows a model card [Mitchell *et al.*, 2019] for each model selected. From the Face-ID *model card*, Divya sees that the model requires images (both for training/registration) and optional ground truth labels.

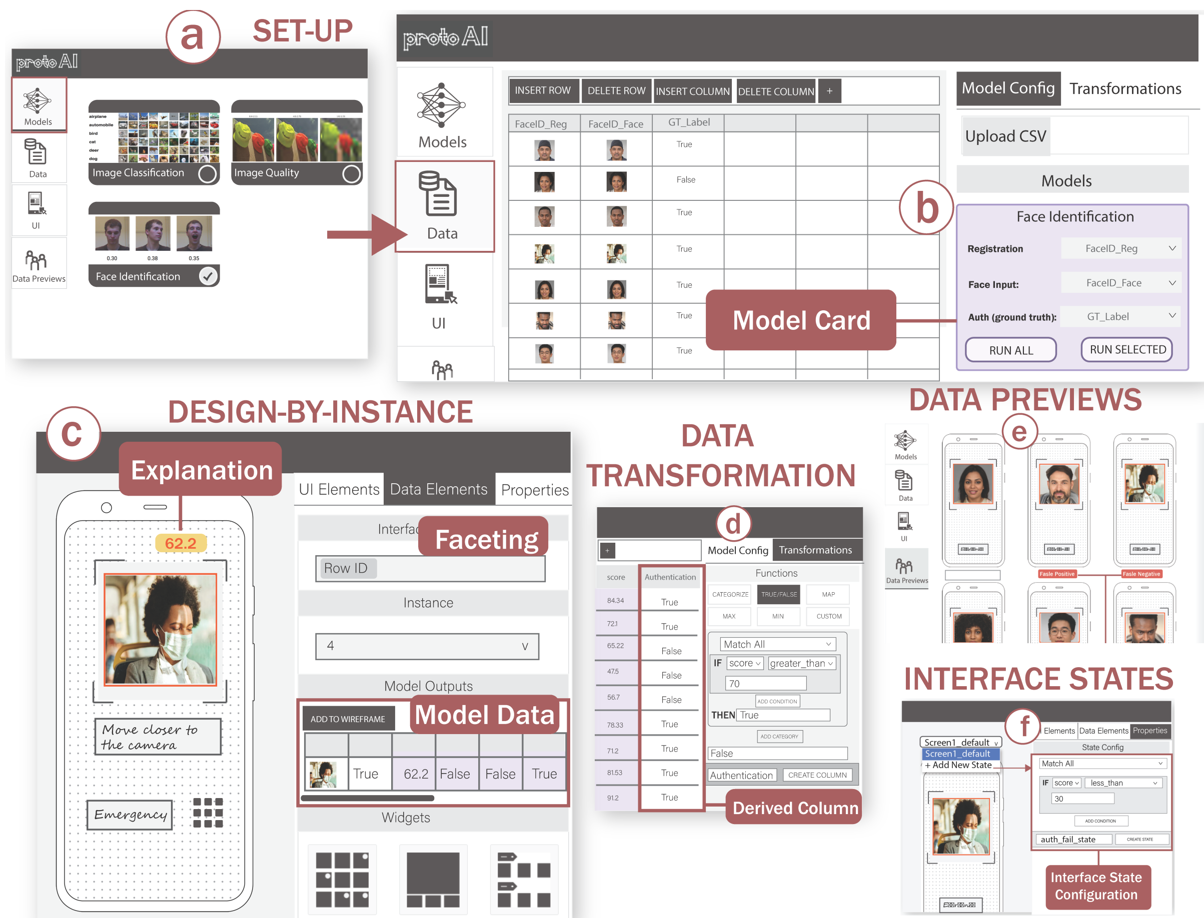


Figure 1: ProtoAI's user interface and features for MIP. To set-up, the designer (a) selects the Face-ID model, and (b) configures it using the model card. In the User Interface tab, the designer (c) incorporates model inputs and outputs in the wireframe; and (d) transforms Face match score into Boolean column in the Data tab.

Based on her user research, Divya has curated a set of personas and portrait photos for each persona taken across different usage context (e.g., low light condition, crowded subway, person with a beard, facial hair, different skin tones, etc.). She uploads this data, configures the Face-ID model card inputs by mapping the column headers and runs the model. ProtoAI extends the spreadsheet and appends additional columns with model outputs. The model output columns are color-coded to match the model configuration card. The Face-ID model that her engineers have created return additional details: a percentage match score (calculated based on face distances in the face embedding space), an explainable heatmap rendering of the input image [Selvaraju *et al.*, 2017], and a set of Boolean flags for model features (e.g., whether a face was detected, eyes were closed, etc.). Using this simulated output, Divya can proceed to design the user interface for Face-ID based unlocking.

## 3.2 User Interface Design

Divya selects the UI tab, which consists of a design canvas and a sidebar for interface elements. The design canvas starts with a default phone template, but Divya can select others if needed (e.g., desktop or tablet). The sidebar consists of three panels, including the *UI Elements* panel which had a set of standard interface elements, the *Data Elements* panel which hosts input and model output data and a collection of widgets for MIP, and a *Properties* panel to set element-specific properties. To design the phone unlock experience, Divya opens the data elements panel (Figure 1c). This panel consists of a faceting control to set the wireframe's *data context* and a table showing the faceted data itself (a subset of the main spreadsheet view). The data context is the scope of end-user data that will be bound to the interface at runtime. From the faceted table, Divya selects the cell value with the persona's face image and clicks on the 'Add to Wireframe' button. ProtoAI adds the image of the person's face onto the template, and Divya can adjust it to fit her design. Divya also adds the percentage match score value from the Face-ID model's output to the interface. While not intended for the final deliverable, Divya can use it to test and debug the interface design. To indicate this to ProtoAI, Divya toggles the 'set as explanation' flag in the properties tab for the score element. This will allow her to selectively show the explanation overviews later in the previews tab. ProtoAI also implements an initial set of widgets for binding Boolean values to images, categorizing items by tags, and showing ranked order of items. Each widget has a predefined layout and can be bound to selected data along with explanation overlays for designers.

### 3.3 Design Evaluation

At this point, Divya has an initial wireframe of the phone unlock interface designed using the portrait image from a single persona. When she selects the Data Previews tab, ProtoAI automatically instantiates the screen interface based on the data context and using all data imported in the data tab (Figure 1e). The Data Previews tab consists of a scrolling grid view of the UI rendered for different users and their portrait photo variations. The preview view allows Divya to rapidly evaluate her design as it is being created and conduct design checks. Divya can also check her design for different data sizes from model output (e.g., recommendations), ranging from no recommendations, a few recommendations, to tens of recommendations. Third, Divya can also evaluate the design for localization by providing inputs in different languages. Fourth, suppose the model's parameters require tweaking (e.g., number of clusters). In that case, Divya can configure the data with different cluster sizes and compare the results in the previews view.

### 3.4 Analysis, Revision, and Repair

ProtoAI's 'evaluation through previews' is intended to support the designer in analyzing design breakdowns in differing real-world contexts. ProtoAI offers a number of analysis features to support this iterative MIP workflow. Because Divya specified ground truth data for each of the photos, ProtoAI automatically compares the ground truth (Face-ID match) with model predictions and tags instances of false positives and false negatives. In the sidebar, ProtoAI provides a summary of each tag indicating the number of instances with that tag. Divya can see that in 16% of data, the model predicted an identity mismatch when the image was, in fact, the persona (i.e., false negatives). By checking the 'show explanations' flag in the sidebar, Divya can see the match score element she added in the UI tab. In this example, Divya sees that the model fails when the person is farther away or when their eyes are closed.

To address this issue, Divya switches back to the Data tab and creates a new calculated Boolean column that is set to 'true' if the face is not detected or when eyes are closed (D3). The Data tab allows for several different types of data *transformations*, including the categorization of numerical values (e.g., high, medium, and low), mapping transformations of model-assigned labels and values to end-user-friendly labels, calculating the minimum and maximum values, and custom formula functions (see Figure 1d). Through these transformations, Divya can control the format in which the model output is presented in the user interface. After creating the Boolean column, Divya returns to the UI tab to address the false-negative instances. In ProtoAI, each screen can be assigned different screen states dependent on model behavior and values. Divya adds a new interface *state* to the unlock screen conditioned on the Boolean column value, which she configures using the properties panel (Figure 1f). In this state, Divya adds a message at the top of the screen prompting the user to move closer to the screen.

## 4 Discussion and Future Work

To design user interfaces for AI-powered applications, designers need access to the underlying AI. Therefore, digital prototyping tools should escape the 'black-box' view of AI by incorporating the AI model's characteristics into the UI prototyping process. In this work, we define a new paradigm for UX design for AI-powered applications, which we call AIX. To accomplish AIX design, we have demonstrated how ProtoAI's implementation of *Model-Informed Prototyping* allows designers to (1) directly incorporate an AI's output into their design, (2) test their design across different input data contexts, and (3) iteratively assess and adapt their interfaces for explainability, failure, and model feedback. This affords opportunities for communication, negotiation, and co-design between designers and engineers. Specifically, future work can investigate how AIX designers can drive AI model parameters based on UI features, negotiate model outputs necessary for explainability, and communicate discovered failure instances with engineers for model improvement.

ProtoAI also has the potential to support Responsible AI needs such as fairness, accessibility, and transparency. AI engineers are asked to evaluate their data and ML models for responsible AI criteria (e.g., AI Fairness 360 [Bellamy *et al.*, 2018]), and AIX designers can use tools like ProtoAI's data previews to detect interface failures in responsible AI design. Finally, as pedagogy and practice of AI application design continues to evolve, we envision AIX tools like ProtoAI will enable students and novice designers to develop necessary skills for AIX prototyping. We imagine a library of widgets implementing AIX design patterns and explainable overlays to scaffold designers' learning process.

## References

[Amershi *et al.*, 2015] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 337–346, 2015.

[Amershi *et al.*, 2019] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 3. ACM, 2019.

[Beaudouin-Lafon and Mackay, 2009] Michel Beaudouin-Lafon and Wendy E Mackay. Prototyping tools and techniques. In *Human-Computer Interaction*, pages 137–160. CRC Press, 2009.

[Bellamy *et al.*, 2018] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[Browne, 2019] Jacob T Browne. Wizard of oz prototyping for machine learning experiences. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2019.

[Dellermann *et al.*, 2019] Dominik Dellermann, Adrian Calma, Nikolaus Lipusch, Thorsten Weber, Sascha Weigel, and Philipp Ebel. The future of human-ai collaboration: a taxonomy of design knowledge for hybrid intelligence systems. 2019.

[Google, 2020] Google. Visually probe the behavior of trained machine learning models, with minimal coding., 2020.

[Hartmann *et al.*, 2006] Björn Hartmann, Scott R Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 299–308, 2006.

[Helms *et al.*, 2018] Karey Helms, Barry Brown, Magnus Sahlgren, and Airi Lampinen. Design methods to investigate user experiences of artificial intelligence. In *2018 AAAI Spring Symposium Series*, 2018.

[Holmquist, 2017] Lars Erik Holmquist. Intelligence on tap: artificial intelligence as a new design material. *interactions*, 24(4):28–33, 2017.

[Horvitz, 1999] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166, 1999.

[Klemmer *et al.*, 2000] Scott R Klemmer, Anoop K Sinha, Jack Chen, James A Landay, Nadeem Aboobaker, and Annie Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10, 2000.

[Maulsby *et al.*, 1993] David Maulsby, Saul Greenberg, and Richard Mander. Prototyping an intelligent agent through wizard of oz. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 277–284, 1993.

[Mitchell *et al.*, 2019] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[Pruitt and Grudin, 2003] John Pruitt and Jonathan Grudin. Personas: practice and theory. In *Proceedings of the 2003 conference on Designing for user experiences*, pages 1–15, 2003.

[Selvaraju *et al.*, 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[Subramonyam *et al.*, 2021] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. Towards a process model for co-creating ai experiences. In *Designing Interactive Systems Conference 2021*, pages 1529–1543, 2021.

[Subramonyam *et al.*, 2022] Hariharan Subramonyam, Jane Im, Colleen Seifert, and Eytan Adar. Solving separation-of-concerns problems in collaborative design of human-ai systems through leaky abstractions. In *CHI Conference on Human Factors in Computing Systems*, pages 1–21, 2022.

[van Allen, 2018] Philip van Allen. Prototyping ways of prototyping ai. *interactions*, 25(6):46–51, 2018.

[Winograd *et al.*, 1986] Terry Winograd, Fernando Flores, and Fernando F Flores. *Understanding computers and cognition: A new foundation for design*. Intellect Books, 1986.

[Wood, 1997] Larry E Wood. *User interface design: Bridging the gap from user requirements to design*. CRC Press, 1997.

[Yang *et al.*, 2019] Qian Yang, Justin Cranshaw, Saleema Amershi, Shamsi T Iqbal, and Jaime Teevan. Sketching nlp: A case study of exploring the right things to design with language intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 185. ACM, 2019.

[Yang *et al.*, 2020] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. Re-examining whether, why, and how human-ai interaction is uniquely difficult to design. In *Proceedings of the 2020 chi conference on human factors in computing systems*, pages 1–13, 2020.