An Infrastructure for Managing AI Bias in Complex Web Systems

Sebastian Heil
Faculty of Computer Science
Chemnitz University of Technology
Chemnitz, Germany

Jannatul Ferdous
Faculty of Computer Science
Chemnitz University of Technology
Chemnitz, Germany

Abubaker Gaber
Faculty of Computer Science
Chemnitz University of Technology
Chemnitz, Germany

Martin Gaedke
Faculty of Computer Science
Chemnitz University of Technology
Chemnitz, Germany

***Abstract*—AI components are increasingly integrated into complex web systems, especially in the context of AI-as-a-Service. However, trustworthiness of the resulting architectures is jeopardized by AI biases in the models used, the data they were trained on, or the way they are deployed. This problem needs to be addressed by systematic approaches of managing the AI bias-related risks. We propose AIBDB, an infrastructure inspired by the Common Vulnerabilities and Exposures (CVE) system that serves as a knowledge base about known AI biases and supports software architects in automatically identifying architectural risks of AI bias. With the first proof-of-concept implementation of this web-based AI bias management infrastructure, we conducted a qualitative user study yielding feedback from 7 participants through thematic analysis of their 112 responses on the usability, usefulness and suggestions for improvement. We contribute to the development of future fair and unbiased AI-driven solutions in the web in the context of the ongoing research on Trustworthy and Human-centric AI and the increasing importance for organizations to address AI bias through recent regulations.**

***Index Terms*—AI Bias, Trustworthy AI, Human-centric AI, Risk Management, AI-as-a-Service, Software Architecture.**

I. Introduction

Artificial Intelligence (AI) has become an integral part of modern web systems (cf. AI-infused systems [1]), supporting applications ranging from personalized recommendations to automated decision-making as organizations are increasingly aware of the potential benefits [2], [3]. AI Integration varies from simple rule-based systems to usage of pre-trained AI-as-a-Service models trained by third parties. Such AI components in web systems can represent speed-ups through automation and add functionalities that would be difficult to implement otherwise [3] and augment human capabilities [2]. The decision to employ AI components and their results quality impacts web system’s users and indirectly, e.g. through damage to reputation and exposure to lawsuits [4], also the organization that operate them [3]. Therefore, *Trustworthy AI* has received much interest in research [5] and policy making [6], [7].

However, biases are becoming an increasingly critical issue in AI systems. Bias can stem from imbalanced training data, flawed algorithmic design, or unintended consequences of deployment, potentially causing unfair or discriminating outcomes. For example, biased AI systems can reinforce stereotypes and existing inequalities, and lead to significant societal harm. AI biases represents a risk for the organization deploying AI components [8], as bias can lead to negative user experiences, loss of trust, and potential legal consequences [4]. Avoiding bias is essential for ensuring the fundamental rights of equality and non-discrimination as basis for building Trustworthy AI systems [6]. Recently, aspects of risk, compliance and data governance are becoming central parts of organizations AI strategies [3], [8].

For these reasons, addressing AI bias in web systems is essential to ensure fairness, transparency, and trust. Security-related risks such as vulnerabilities and exposures, in contrast, are well understood and have been extensively studied, which lead to the creation of systematic approaches for their management. However, the management of AI bias is still an newly evolving field, with many open challenges remaining. Futhermore, as web systems increasingly influence societal and individual decisions, managing AI bias in them is not just a technical challenge but also of high ethical and societal relevance. Recent legislations like the EU AI Act [7] and the Algorithmic Accountability Act [9] in the US highlight the increasing need for organizations to address bias in AI systems.

This paper aims at exploring management of AI bias in the design of complex web systems. We take inspiration from the well-established field of security risk management and propose a systematic approach to capture bias information in a public web-based knowledge repository similar to the Common Vulnerabilities and Exposures (CVE) system [10]. Our novel approach specifies a software architecture and

This work is supported by the EU’s HORIZON Research and Innovation Programme under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI).

data model for representing known AI biases, which enables software architects to assess the risk of AI bias in their web system designs and get information about potential mitigation strategies. By this, we seek to contribute to the development of future fair and unbiased AI-driven solutions in the web.

The remainder of this paper is structured as follows: Section II reviews related work and background on AI bias and risk management. Section III presents our proposed architecture and data model for an AI bias registry. Section IV discusses the evaluation of our approach through illustrative scenarios. Finally, Section V concludes the paper and outlines directions for future work.

II. Background and Related Work

AI bias refers to the systematic and unfair discrimination against certain groups or individuals in AI systems. It can be introduced in a system both intendedly and unintendedly and represents a complex challenge due to socio-technical nature of AI systems and their potential to interfere with legally protected attributes such as race, gender, religion or disabilities [8]. In the following, we provide an overview on the research on AI bias detection/mitigation and management, and position our work in relation to that.

Bias Detection and Mitigation Techniques. A strong focus of current research in addressing AI bias is on bias detection. Various methods have been proposed to detect bias in AI systems, including statistical tests, adversarial testing, and interpretability techniques [11].

Statistical tests assess the fairness of predictions across different groups (e.g. demographic minority groups), while adversarial testing involves the creation of scenarios that can reveal biases in model behavior. Alelyani et al. [12] propose a method for bias detection in machine learning models that alternates values of potentially biased features and observes the impact of these variations on the predictions. In contrast, outcome testing allows to detect biases without requiring prior knowledge of the model's internal workings or relevant features, by comparing the outcomes of different groups based on the concept of predictive parity [13]. Benchmarking the result distributions based on the distribution of sensitive features like race or gender is another alternative to detect biases [13].

Adversarial testing can be used to address dataset biases through iterative re-training and aggressively filtering oversampled counterfactuals [14] or a more bottom-up model-based approach removing spurious artifacts in the data [15].

Interpretability and explainable AI (XAI) techniques are a different approach to bias detection, as they make the often hidden decision-making criteria of AI systems visible. For instance, it has been shown that the application of SHAP (SHapley Additive exPlanations) can support the detection of structural bias in convolutional NN models [16].

First efforts can be observed to consolidate bias detection and mitigation algorithms in one place in order to facilitate practitioners to assess and address bias in their AI systems, such as in AIF360 [17]. Unlike the bias detection techniques outlined above, our approach focuses on the systematic management of bias-related information in complex web systems, enabling software architects to assess and mitigate risks from AI bias in their designs with regard to already known biases.

AI Bias Management. Management of AI bias has not yet received similar attention like its detection and mitigation. However, managing AI bias is an crucial aspect of the wider topic of AI *risk management* aimed at minimizing the "anticipated and emergent negative impacts of AI systems" [8]. Roselli et al. propose to combine "quantitative assessments, business processes, monitoring, data review, evaluations and controlled experiments" [4] for a wholistic approach of managing bias. They particularly point out that with increasing use of AI models from sources outside the organizations employing them (cf. *AI-as-a-Service*), bias management processes should not require a deep understanding of AI models' inner workings in order to be applicable in industry scenarios with non-technical management stakeholders, auditors or regulators [4]. Our AI bias management solution follows the same rationale in treating AI components as blackboxes with only metadata like the model identifier available, and the supporting infrastructure we propose in section III is intended as a "targeted toolset" [4] for the evaluation activity proposed by Roselli et al.

Consideration of AI bias is particularly important at the early stages of the AI lifecycle, i.e. in the Pre-Design and Design & Development stage where the selection of datasets and models can have a strong impact [8]. A *Human-centered Design process* and ensuring human oversight are important for managing AI bias related risks, where a "*risk mitigation mindset*" — that means accepting that biases can occur and need to be mitigated quickly — is crucial [8]. In this context, our proposed solution is a contribution to facilitate quick awareness and reaction to the presence of biases in complex web systems once they become known. Transparency is a key objective for managing bias and has recently been addressed by proposals like Model Cards [18].

The AI Incident Database (AIIDB) [19] pursues a similar objective by focusing on *information sharing* [8] about AI-related incidents across stakeholders. While not specifically targeting AI biases, it takes inspiration from the NTSB aviation incident database and the CVE system. The platform we propose in section III represents a database of known AI biases with a dedicated focus on their technical representation and interfaces that enable integration with architectural modeling tools to support automatic identification and notification use cases for software architects.

III. AI Bias Management with AIBDB

In this section, we present our solution to facilitate AI bias management for Software Architects of web-based systems. To that end, we propose a novel software architecture – AIBDB– that specifies a platform for capturing known AI bias information with a focus on enabling support for automatic detection and notification about AI bias related risks in architectural models of web-based systems. Figure 1 provides an overview on the architecture of AIBDB, its main components and their

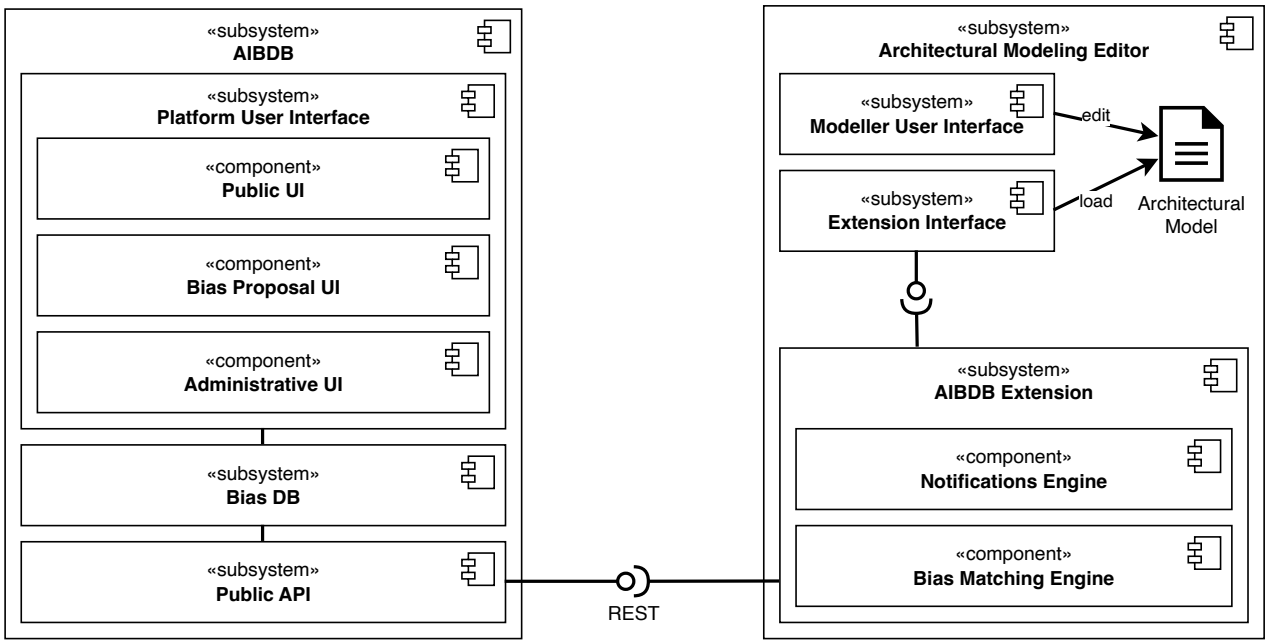

Fig. 1. Architecture of the AIBDB infrastructure

interactions with each other as well as its integration into an architectural modeling solution to enable notification and support functionality for software architects. The two-sided nature of the figure reflects the two main use cases of AIBDB: (i) the left side shows the interaction of users with the AIBDB platform to search for known AI biases and submit new bias reports, and (ii) the right side shows the integration of AIBDB with architectural modeling tools to enable automatic detection and notification of potential bias risks in architectural models. The target audience is divided into two groups likewise: (i) direct users of the AIBDB platform, who are interested in searching for known AI biases and submitting new bias reports, and (ii) software architects, who are interested in assessing the risk of bias in their architectural models and receiving notifications about potential bias risks. Similar to the CVE system [10], the AIBDB platform is intended to be used by a wide range of stakeholders, including software architects, data scientists, and AI researchers, as well as organizations and regulatory bodies interested in monitoring and managing AI bias, and the direct platform users create the knowledge base of known AI biases that is then used by the software architects using a community-driven/crowdsourced approach.

Therefore, the AIBDB platform is designed to be a public web-based knowledge repository for known AI biases, similar to the CVE system [10]. The AIBDB architecture consists of three main components: the `Platform User Interface`, the `Bias DB`, and the `Public API`. The `Platform User Interface` is the main entry point for users to interact with the AIBDB platform itself. It provides a web-based interface for users to search for known AI biases, view detailed information about them, and submit new bias reports. The `Bias DB` is the core component of the AIBDB architecture, where all known AI biases are stored. We describe the data model of the `Bias DB` in detail in section III-A. The `Public API` provides an interface for external systems to interact with the AIBDB platform, in particular to enable the automatic notification of software architects about potential bias risks in their architectural models through integration with architectural modeling tools.

A. Data Model of the AIBDB Platform

To support the above functionalities, a data model capturing the relevant information about known AI biases is required. At the core of the data model is a list of `Bias` entities, each representing a known AI bias. Each `Bias` entity has a unique identifier (id) and a set of further attributes that describe the bias in detail. The format of the identifier follows the pattern of the CVE system [10] as follows:

`<Prefix>-<Year>-<Sequential Number>`

in which:

- `Prefix` is the constant string `AIB` to indicate that it is an AIBDB AI bias identifier,
- `Year` is the year of the bias submission to encode recency information in the identifier itself and providing a namespace-like structure limiting the required length of the Sequential Number, and
- `Sequential Number` is an increasing number assigned to the bias within that year with a length of four digits[1].

[1]Four digits are currently used for CVE identifiers, allowing for 10k CVEs/AI Biases per year

The `type` attribute indicates the specific type of the bias itself – not to be confused with the type of the bias source, which is addressed below – and can take various types such as *gender*, *racial*, *social* etc. The `description` attribute provides a human-readable representation of the bias, including its source and potential mitigation strategies and should be written in a way that is understandable for non-AI-experts. The content of this attribute is used for displaying the bias information in the `Platform User Interface` as well as in systems that integrate with the platform such as architectural modeling tools in order to support software architects in dealing with potential bias risks in their designs. The `applies-to` attribute connects a `Bias` entity to `FilterSpecification` that aggregates information relevant for the matching of known AI biases with the architectural models of web systems.

A `FilterSpecification` has an attribute `source-type` that indicates type of the the source of the bias and can take one of the following values: `source-type` $\in \{dataset, algorithm\}$. The `source` attribute is a reference to the bias source which, depending on the `source-type`, points to either a `Dataset` or an `Algorithm` entity. As both Datasets and Algorithms evolve over time, the FilterSpecification comprises of two additional attributes: `versions` and `published`. The `versions` attribute is applicable for Datasets and Algorithms which use a semantic versioning scheme[2] to distinguish different releases and it can use a syntax similar to the one used in Python package management[3]. This allows to specifify a single version, e.g. `1.0`, as well as a minimum or maximum version, e.g. `<3.0` or `>=1.0`, or specific range of versions, e.g. `>=1.0,<3.0`. Alternatively, the `published` attribute can be used to specify a date range in which the bias is applicable. We propose to use the ISO ISO 8601-1:2019 date format[4] for the `published` attribute, which allows to specify a single date to refer to a point in time at which the Dataset or Algorithm was published, e.g. `2025-05-05`, or a date range, e.g. `2023-10-01/2024-10-01`.

The `Dataset` and `Algorithm` entities are used to represent the specific sources of the AI biases. Both have a name attribute that provides a human-readable name for displaying them, as they are used to enable autocompletion/selection of already known Datasets and Algorithms in the `Platform User Interface` when submitting a new bias proposal.

`Bias`, `Dataset` and `Algorithm` entities connected to potentially several `Reference` entities that provide additional information about the bias. These references can either be in the form of a citation to a scientific paper or a URL to a website that provides more information about the bias, dataset or model, using the corresponding attributes.

The AIBDB data model is designed to be extensible, allowing to add new bias types, algorithms or datasets on the instance data level without requiring changes to the underlying data model structure.

[2]cf. https://semver.org/

[3]cf. https://pip.pypa.io/en/stable/user_guide/

[4]cf. https://www.iso.org/standard/70907.html

B. Integration with Architectural Modeling Tools

To facilitate identification of known AI biases present in web system designs, AIBDB integrates with architectural modeling tooling. Conceptually, this is achieved through the provision of a REST interface exposing the public API of the AIBDB platform and the integration into an modeling editor as an extension. Using the information about known AI biases from the AIBDB platform, this extension checks an `Architectural Model` for matching characteristics through a `Bias Matching Engine`. If biases are found to be present in the Architectural Model, the `Notifications Engine` alerts the Software Architect who initiated the check.

The `Bias Matching Engine` identifies known biases by looking for the bias identification patterns expressed in the `FilterSpecification` of the AIBDB data model (cf. section III-A. Therefore, the architectural model fulfills a similar function like a software inventory when checking for CVEs [10], providing an information source that represents AI Algorithms and Datasets involved in the web system's design.

The `Notifications Engine` is responsible for displaying information about the known AI biases found in the architectural model. For this, it needs to visually highlight affected components of the system architecture and modify viewport and zoom level to ensure visibility in complex and large architectural models. It is also responsible for displaying additional information about the biases as represented in the description attribute in the AIBDB data model.

Our proof-of-concept (PoC) implementation showcases this integration with an existing editor for WAM (WebComposition Architecture Model), as WAM represents a visual architectural modeling language for federated web systems which has been extended to address security and authorization aspects and that defines an ontological mapping allowing to represent the modeled architectures as RDF-based knowledge graphs [20], providing a solid basis for the querying required for the `Bias Matching Engine` and a high level of interoperability due to the mature ecosystem of the Semantic Web. **??** shows an example of found biases in the AIBDB extension.

IV. Evaluation

For a pilot evaluation of AIBDB, we performed empirical experimentation with the PoC implementation. The evaluation objective is to get a first feedback on the usability and usefulness of AIBDB in supporting bias-aware design of web systems. To that end, we chose to employ a qualitative methodology in the form of a user study that consisted of task-based interactions with AIBDB followed by structured post-task interviews. This research design allowed us to pursue an exploratory user-centric approach to understand how engineers use the features of AIBDB and what supports or impedes them in the main use cases of *Checking* and *Information*

A. Material

Three main materials were used in the experimental evaluation: 1) the proof-of-concept implementation of AIBDB, 2)

the set of scenario tasks the participants had to perform, 3) the post-task questionnaire. All materials are available along with the evaluation data as part of the replication package[5].

The PoC implementation of AIBDB consists of two parts: the AIBDB platform itself and the integration into an existing WAM editor. The platform was implemented using JavaScript, using Node.js with Express.js for the platform backend and REST API. The AIBDB frontend is developed component-based using React. The data model presented in section III-A was realized using PostgreSQL. Integration with the WAMflow was implemented in Next.js Server-side rendering (SSR). For the experimentation, the implementation was hosted on Vercel, deploying the backend on Render and the database on Supabase. These running versions of the AIBDB platform[6] and the integration in WAMflow[7] are available for review.

Two scenario tasks were designed for the empirical experimentation. Task T1 represents interactions with the platform user interface for the use cases of AI Bias Submission and Information.It comprises submitting a new bias to the platform and using the search and filter features to look for information regarding an existing AI bias. Task T2 represents the Checking use case of the AIBDB extension of the WAMflow architectural modeling editor. It comprises the creation of a WAM model with an AI component and a dataset which are subject to known biases and the use of the detection capabilities to get notified and review the bias details.

The post-task questionnaire contains a demographics section (age, gender, occupation, professional experience) followed by the 6 specific modules with 16 open-ended questions and was adminstered via Google Forms. The 6 modules for the qualitative data collection were focussed on the general User Experience (2 questions), usability wrt. interface and navigation (5), usefulness of the overall solution (3), workflow and integration (2), required improvements (2) and further recommendations and comments (2).

B. Procedure

The experimental procedure is based on the six-phase methodology for thematic analysis by Braun and Clarke [21]. For the raw data collection, after introduction to the experiment and consent to participate, each of the participants was performing both tasks based on the task descriptions. About 10-15 minutes were needed for task T1 and 15-20 minutes for task T2 due to its slightly higher complexity. After task completion, the post-task questionnaire was used to capture the participant's perceptions and opinions about AIBDB. After initial familiarization with the collected participant responses, the initial coding was performed. Resulting codes were then clustered and organized into overarching themes that were then re-checked against the responses for consistency and relevance, resulting in refined themes. Then, naming and definition of the refined themes was performed and a report with illustrative quotes was created.

[5]anonymized for review: https://figshare.com/s/04be04d7cbb190c38235

[6]https://bias-management-system.vercel.app/, temp. login admin:admin123

[7]https://wameditor.vercel.app/homePage

C. Results

Orienting on Nielsen's rule of thumb for qualitative usability studies [22], we recruited 7 participants (4 male,3 female) who performed the tasks of the user study outlined above between May 21 and June 3, 2025, collecting a total of 112 responses. The age distribution of our participants is as follows: 18-24 years: 1 participant, 25-34 years: 3 participants, 35-44 years: 2 participants, 45-54 years: 1 participant. 2 participants self-qualified themselves as Master students, 1 as PhD student, 2 as Web Developers and 2 as Software Engineers. In terms of professional experience, 1 participant was in the range up to 1 year, 4 had up to 3 years, 1 up to 6 years and 1 up to 10 years. Across all participant responses, we have identified the 6 recurring themes.The detailed results from our experimentation are available in the replication package[6].

D. Discussion

Findings by Themes. In the following, we briefly outline the findings in the 6 identified themes.

Interface Intuitiveness/Clarity was seen mostly positive, as the majority of participants found UI layout and design logic intuitive and goal-oriented, highlighting the clear separation between the modeling pane and the metadata panel as a strength. Expectedly, especially participants with lower previous experience found the architectural modeling itself difficult.

Functional Comprehensiveness/ Workflow Support was generally perceived positive. Likewise, all participants were able to complete the tasks without major interaction problems. In particular, the simplicity of the bias detection feature was praised. For the bias detection result list, an improved filtering was suggested when dealing with larger models. This was also the most frequent of all suggestions.

Visual Feedback/ Navigation Limitations. The coded segments belonging to this theme focused on the highlighting of architectural components with known biases, which was well-received for simpler models. However, more complex architectures were considered more difficult due to the automatic zooming feature requiring manual re-adjustments to improve context understanding. Similarly, the understanding of the clickable zones in an architectural model was challenging for some users. These feedbacks highlight the importance for further improvements of the notification mechanism and interaction with the model in the modeling pane of the editor.

Metadata Structure/ Traceability feedback concentrated on clarity of the AIBDB data model as represented in the property pane of the editor and the platform user interface, indicating a good understandability and clarity of the corresponding attributes. Suggestions for improvement comprised addition of a search/filter functionality for the editable metadata attributes.

User Responsibility/Role as recurring theme in the feedback shows a good understanding of the different capabilities depending on the user role. We did not record negative feedback regarding this theme.

Improvement was the theme in several user responses. In addition to the aspects mentioned for the specific themes above, participants also suggested adding a guidance for new

users not familiar with AIBDB, especially for users with lower architectural modeling expertise.

Threats to Validity. *Internal validity* of the presented pilot evaluation is primarily limited by the convenience sampling of the participants and the coding procedure. We addressed this by ensuring the same neutral introduction was given to all participants and the questionnaire was followed and presented in the same way, trying to minimize bias. For the thematic analysis, we followed the established guidelines by Braun and Clarke [21] and had one researcher conduct the analysis with a second researcher verifying.

External validity is limited by the study design and the demographics of the participants. While the total number of 7 participants is sufficient and common for qualitative evaluations, a larger-scale quantitative/mixed-methods evaluation of the proposed approach is required with the improved version based on the qualitative feedback gathered in this study. For that reason, we are not making any claims towards the representativeness of the overall positive feedback from the participants. Generalization of our experimental results beyond the presented analysis is not intended andas the primary purpose was to identify problems and areas for improvement.

Construct validity is threatened by the design of the questionnaire. We designed it to cover a wide range of aspects of interest in the 6 modules and open-ended questions allowing participants to bring up interaction problems and suggestions unforseen by us. For coding consistency, we used the two-role verification process mentioned above.

V. Conclusion

In this paper, we proposed a novel approach for managing AI bias in architectural designs of web system inspired by CVEs, comprising of a software architecture and data model for representing known AI biases. It enables software architects to assess the risk of AI bias in their web system designs, contributing to the development of future fair and unbiased AI-driven web solutions. Unlike existing approaches, we address identification of known biases at the design stage. In a qualitative pilot study, we gained insights in the applicability, limitations and areas for improvement of AIBDB. All implementation and experimental materials are provided to allow to replicate our experiments and extend the approach.

Future work aims at transforming the experimental feedback into an improved version of the platform and the integration with further architectural modeling tools and the combination with model cards [18]. The updated version is planned to be evaluated in a larger-scale user study with industry partners in the context of project anonymized . Possible future extensions of our approach would be an integration with AIIDB or the CVE system.

References

[1] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, "Guidelines for Human-AI Interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, may 2019, pp. 1–13.

[2] M. H. Jarrahi, "Artificial intelligence and the future of work: Human-AI symbiosis in organizational decision making," *Business Horizons*, vol. 61, no. 4, pp. 577–586, jul 2018.

[3] A. Singla, A. Sukharevsky, L. Yee, and M. Chui, "The State of AI: How organizations are rewiring to capture value." [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

[4] D. Roselli, J. Matthews, and N. Talagala, "Managing Bias in AI," in *Companion Proceedings of The 2019 World Wide Web Conference*. New York, NY, USA: ACM, may 2019, pp. 539–544.

[5] B. Li *et al.*, "Trustworthy AI: From Principles to Practices," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–46, sep 2023.

[6] AI High-Level Experts Group (AI HLEG), "Ethics Guidelines for Trustworthy AI," European Commission, Tech. Rep., 2019. [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[7] European Parliament and Council, "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)," *Official Journal of the European Union*, 2024.

[8] R. Schwartz, A. Vassilev, K. Greene, L. Perine, A. Burt, and P. Hall, "Towards a standard for identifying and managing bias in artificial intelligence," National Institute of Standards and Technology, Tech. Rep., mar 2022. [Online]. Available: https://doi.org/10.6028/NIST.SP.1270

[9] House of Representatives, Congress. H.R. 6580 (IH), "Algorithmic Accountability Act of 2023," 2023. [Online]. Available: https://www.congress.gov/bill/117th-congress/house-bill/6580

[10] The MITRE Corporation, "CVE - Common Vulnerabilities and Exposures," 2025. [Online]. Available: https://www.cve.org

[11] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and machine learning: limitations and opportunities*. The MIT Press, 2023.

[12] S. Alelyani, "Detection and Evaluation of Machine Learning Bias," *Applied Sciences 2021, Vol. 11, Page 6271*, vol. 11, no. 14, p. 6271, jul 2021.

[13] R. Fu, Y. Huang, and P. V. Singh, "AI and Algorithmic Bias: Source, Detection, Mitigation and Implications," *SSRN Electronic Journal*, jul 2020.

[14] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, "SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, pp. 93–104.

[15] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "WinoGrande: an adversarial winograd schema challenge at scale," *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, sep 2021.

[16] B. Van Stein, D. Vermetten, F. Caraffini, and A. V. Kononova, "Deep BIAS: Detecting Structural Bias using Explainable AI," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, jul 2023, pp. 455–458.

[17] R. K. E. Bellamy *et al.*, "AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias," Oct. 2018. [Online]. Available: https://arxiv.org/abs/1810.01943

[18] M. Mitchell *et al.*, "Model cards for model reporting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*. New York, NY, USA: ACM, jan 2019, pp. 220–229.

[19] S. McGregor, "Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 15 458—-15 463.

[20] A. Scholtz, S. Wild, and M. Gaedke, "Systematic Composition of Web-based Applications with Focus on Security," *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, 2015.

[21] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, jan 2006.

[22] J. Nielsen, "How many test users in a usability study? - nn/g," 2012. [Online]. Available: https://www.nngroup.com/articles/how-many-test-users/