CS59200-ASE AI-Assisted Software Engineering

Instructor: Tianyi Zhang, Assistant Professor of Computer Science

Email: tianyi@purdue.edu

Lecture: 12:00pm-1:15pm TTh @ LWSN B134

Office Hours: 1:30pm-2:30pm TTh @ LWSN 3154H

Instructional Modality: Face-to-Face

Course Credits: 3.0

Prerequisites: Python programming skills and basic understanding of machine learning are required. Knowing how to use scikit-learn and PyTorch (or Keras) is recommended.

Course Description

Have you ever wondered about these:

- Will programming jobs no longer exist because of large language models like ChatGPT?[1]
- Will software engineering become prompt engineering in the next decade?
- How far are we from the “black art” of natural language programming as Dijkstra called it 40 years ago?[2]
- What essential software engineering skills are needed in the age of AI?

This course will help you answer those questions. In this course, you will learn concepts and research topics about (1) the role of human programmers in the age of AI, (2) how AI and ML technologies have been applied in different software engineering tasks such as code generation and software testing, (3) how AI-based software development tools work with or clash against the existing workflow of programmers, and (4) how to improve AI-based software development tools for better performance, robustness, and usability. While there has been a long history of applying Artificial Intelligence and Machine Learning to software engineering tasks[3], we will focus on Deep Learning models, especially the recent advances in Large Language Models, in this course.

From Week 1 to Week 4, the instructor will give lectures on ML basics and deep learning. In the following weeks, students will take turns to present and discuss research papers.

---

[1] Matt Welsh, The end of programming. *Communications of the ACM*, January 2023.

[2] E. W. Dijkstra. On the foolishness of “natural language programming". In *Program Construction*, pages 51–53. Springer, 1979.

[3] Richard J Waldinger and Richard CT Lee. PROW: A Step toward Automatic Program Writing. IJCAI, 1969.

You should expect to present one research paper during the semester. You also need to write a short paper review (2 or 3 paragraphs) in the form of comments and questions for each paper in this course, no matter you are the presenter or not. There will be a course project, in which you will work in groups to design and carry out research projects related to AI-assisted software engineering.

Learning Resources, Technology, Texts

Since this is a relatively new research topic, there is currently no textbook available for this course. I recommend you reading some surveys to get a basic understanding about the literature.

- Yang, Yanming, Xin Xia, David Lo, and John Grundy. "A survey on deep learning for software engineering." ACM Computing Surveys (CSUR) 54, no. 10s (2022): 1-73.
- Zan, Daoguang, Bei Chen, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Wang Yongji, and Jian-Guang Lou. "Large language models meet NL2Code: A survey." In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 7443-7464. 2023.
- Fan, Angela, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. "Large language models for software engineering: Survey and open problems." arXiv preprint arXiv:2310.03533 (2023).

If you are not very familiar with Machine Learning or Deep Learning. I recommend reading the following textbooks.

- Christopher M. Bishop (2006), Pattern Recognition and Machine Learning
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning

To develop useful AI-assisted programming tools, it is very important to understand user interface design and Human-AI Interaction. I recommend read the following articles:

- Jakob Nielson, 10 Usability Heuristics for User Interface Design
- Google PAIR, People + AI Guidebook
- Apple, Human Interface Guidelines for Machine Learning
- Microsoft, HAX Toolkit

We will use Ed Discussion to make announcements, ask & answer questions, look for teammates, etc. If you have not been added to this course on Ed Discussion, please enroll via this link https://edstem.org/us/join/pFbpDJ.

All lecture slides, assignment instructions, and other materials will be posted on Purdue BrightSpace.

Learning Outcomes

At the end of this course, students should be able to:

- design and implement AI-based software engineering tools
- develop effective mechanisms to improve the performance, robustness, and usability of your tools
- evaluate an AI-based software engineering tool through quantitative experiments and user studies
- assess the strengths and weaknesses of a research idea or paper
- write a research paper and give research presentations

Grading

Reading assignments [20%]
Paper presentation [10%]
Course project [55%]
- Project proposal [5%]
- Midterm project report [10%]
- Final project report [30%]
- Final presentation and video demo [10%]

Pop Quizzes [10%, 2% each quiz]
Class participation and discussion [5%]

Course Schedule

Week 1. Introduction to AI-assisted Software Engineering	
Tue 1/9	[Lecture] Course Overview Optional Reading: • Matt Welsh, The End of Programming (CACM)
Thu 1/11	[Lecture] Introduction to AI-assisted Software Engineering Optional Reading: • Yang et al., A Survey on Deep Learning for Software Engineering (CSUR 2022) • Fan et al., Large language models for software engineering: Survey and open problems (arXiv 2023)

Week 2: Machine Learning Basics	
Tue 1/16	[Lecture] Probability Theory and Linear Algebra Optional Reading: • Pattern Recognition and Machine Learning, Chapter 1 and Chapter 2
Thu 1/18	[Lecture] MLE, Loss Functions, and Gradient Descent Optional Reading: • Principles of Data Mining, Chapter 4.5.2 • Principles of Data Mining, Chapter 8.3
Sample course project ideas released by the instructor on 1/21	
Week 3: Deep Learning Models	
Tue 1/23	[Lecture] Neural Networks and Backpropagation Optional Reading: • Pattern Recognition and Machine Learning, Chapter 5
Thu 1/25	[Lecture] Popular Neural Network Architectures Optional Reading: • Deep Learning, Chapters 9, 10
Week 4. Transformers and Large Language Models	
Tue 1/30	[Lecture] Attention Mechanisms and Transformers Optional Reading: • Transformers for Machine Learning: A Deep Dive, Chapter 2
Thu 2/1	[Lecture] Large Language Models Optional Reading: • Zhao et al., A Survey of Large Language Models (arXiv 2023)
Week 5. Representation Learning for Code	
Tue 2/6	[Student Presentation] Alon et al., code2vec: Learning Distributed Representations of Code (POPL 2019) [Student Presentation] Feng et al., CodeBERT: A Pre-Trained Model for Programming and Natural Languages (EMNLP Findings 2020)

Thu 2/8	[Student Presentation] Duo et al., GraphCodeBERT: Pre-training Code Representations with Data Flow (ICLR 2021) [Student Presentation] Niu et al., SPT-Code: Sequence-to-Sequence Pre-Training for Learning Source Code Representations (ICSE 2022)
Week 6. Machine Learning for Code Generation I	
Tue 2/13	[Student Presentation] Yin and Neubig, A Syntactic Neural Model for General-Purpose Code Generation (ACL 2017) [Student Presentation] Svyatkovskiy et al., IntelliCode Compose: Code Generation using Transformer (ESEC/FSE 2020)
Thu 2/15	[Student Presentation] Chen et al., Evaluating Large Language Models Trained on Code (arXiv 2021, the Codex paper) [Student Presentation] Zhang et al., RepoCoder: Repository-Level Code Completion Through Iterative Retrieval and Generation (EMNLP 2023)
Project proposal due on 2/18 midnight	
Week 7. Machine Learning for Code Generation II	
Tue 2/20	[Student Presentation] Xu et al., Incorporating External Knowledge through Pre-training for Natural Language to Code Generation (ACL 2020) [Student Presentation] Chen et al., CodeT: Code Generation with Generated Tests (ICLR 2023)
Thu 2/22	[Student Presentation] Chen et al., Teaching Large Language Models to Self-Debug (arXiv 2023) [Student Presentation] Guo et al., Learning to Complete Code with Sketches (ICLR 2022)
Week 8. Machine Learning for Software Testing and Vulnerability Detection	
Tue 2/27	[Student Presentation] Schafer et al., An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation (TSE 2023) [Student Presentation] Liu et al., Fill in the Blank: Context-aware Automated Text Input Generation for Mobile GUI Testing (ICSE 2023)
Thu 2/29	[Student Presentation] Zhou et al., Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks (NeurIPS 2019)

	[Student Presentation] Pearce et al., Examining Zero-Shot Vulnerability Repair with Large Language Models (SP 2023)
Week 9. Machine Learning for Fault Localization and Program Repair	
Tue 3/5	[Student Presentation] Li et al., DeepFL: Integrating Multiple Fault Diagnosis Dimensions for Deep Fault Localization (ISSTA 2019) [Student Presentation] Yang et al., Large Language Models for Test-Free Fault Localization (ICSE 2024)
Thu 3/7	[Student Presentation] Jiang et al., CURE: Code-Aware Neural Machine Translation for Automatic Program Repair (ICSE 2021) [Student Presentation] Xia et al., Automated Program Repair in the Era of Large Pre-trained Language Models (ICSE 2023)
Week 10. No Class (Spring Break)	
Week 11. Machine Learning for Requirements Engineering	
Tue 3/19	[Student Presentation] Di Sorbo et al., What Would Users Change in My App? Summarizing App Reviews for Recommending Software Changes (FSE 2016) [Student Presentation] Villarroel et al., Release Planning of Mobile Apps Based on User Reviews (ICSE 2016)
Thu 3/21	[Student Presentation] Wang et al., Where is Your App Frustrating Users? (ICSE 2022) [Student Presentation] Luo et al., PRCBERT: Prompt Learning for Requirement Classification using BERT-based Pretrained Language Models (ASE 2022)
Mid-point project summary due on 3/24 midnight	
Week 12. Machine Learning for Software Documentation	
Tue 3/26	[Student Presentation] LeClair et al., A Neural Model for Generating Natural Language Summaries of Program Subroutines (ICSE 2019) [Student Presentation] Xu et al., Commit Message Generation for Source Code Changes (IJCAI 2019)
Thu 3/29	[Student Presentation] Geng et al., Large Language Models are Few-Shot Summarizers: Multi-Intent Comment Generation via In-Context Learning (ICSE 2024) [Student Presentation] Yang et al., APIDocBooster: An Extract-Then-Abstract Framework Leveraging Large Language Models for Augmenting API Documentation (arXiv 2023)

Week 13. Machine Learning for Software Maintenance	
Tue 4/2	[Student Presentation] Chakraborty et al., CODIT: Code Editing with Tree-Based Neural Models (TSE 2020) [Student Presentation] Dilhara et al., PyEvolve: Automating Frequent Code Changes in Python ML Systems (ICSE 2023)
Thu 4/4	[Student Presentation] Kang et al., Explainable Automated Debugging via Large Language Model-driven Scientific Debugging (arXiv 2023) [Student Presentation] Tufano et al., Using Pre-trained Models to Boost Code Review Automation (ICSE 2022)
Week 14. Explainable AI for Code Models	
Tue 4/9	[Student Presentation] Pornprasit et al., PyExplainer: Explaining the Predictions of Just-In-Time Defect Models (ASE 2021) [Student Presentation] Paltenghi and Pradel, Thinking Like a Developer? Comparing the Attention of Humans with Neural Models of Code (ASE 2021)
Thu 4/11	[Student Presentation] Yang et al., What Do Code Models Memorize? An Empirical Study on Large Language Models of Code (ICSE 2024) [Student Presentation] Zhou et al., CoLeFunDa: Explainable Silent Vulnerability Fix Identification (ICSE 2023)
Week 15. Open Challenges in AI-based Software Engineering	
Tue 4/16	[Lecture] Open Challenges in AI-based Software Engineering Part I Optional Reading: • Vaithilingam et al., Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models (CHI 2022 LBW) • Ross et al., The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development (IUI 2023) • Mozannar et al., Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming (arXiv 2022) • Wang et al., Practitioners' Expectations on Code Completion (arXiv 2023) • Barke et al., Grounded Copilot: How Programmers Interact with Code-Generating Models (OOPSLA 2023) • Liu et al., On the Reliability and Explainability of Language Models for Program Generation (TOSEM 2024) • Zhuo et al., On Robustness of Prompt-based Semantic Parsing with Large Pre-trained Language Model: An Empirical Study on Codex (EACL 2023) • Asare et al., Is GitHub's Copilot as Bad as Humans at Introducing Vulnerabilities

	in Code? (EMSE 2024) • Pearce et al., Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions (SP 2022) • Siddiq et al., An Empirical Study of Code Smells in Transformer-based Code Generation Techniques (SCAM 2022) • Cineselli et al., To What Extent do Deep Learning-based Code Recommenders Generate Predictions by Cloning Code from the Training Set? (MSR 2022) • Wang et al., Natural Language to Code: How Far Are We? (ESCE/FSE 2023)
Thu 4/18	[Lecture] Open Challenges in AI-based Software Engineering Part II Optional Reading: • Noller et al., Trust Enhancement Issues in Program Repair (ICSE 2022) • Kabir et al., Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions (arXiv 2023) • Yuan et al., No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation (arXiv 2023) • Le and Zhang, Log-based Anomaly Detection with Deep Learning: How Far Are We? (ICSE 2022) • Zhou et al., Generation-based Code Review Automation: How Far Are We? (ICPC 2023) • Guo et al., Exploring the Potential of ChatGPT in Automated Code Refinement: An Empirical Study (ICSE 2024) • Chakraborty et al., Deep Learning Based Vulnerability Detection: Are We There Yet? (TSE 2022) • Zeng et al., An Extensive Study on Pre-trained Models for Program Understanding and Generation (ISSTA 2022) • Zhu et al., Deep is Beter? An Empirical Comparison of Information Retrieval and Deep Learning Approaches to Code Summarization (TSE 2023) • Hadi and Fard, Evaluating pre-trained models for user feedback analysis in software engineering: A study on classification of app-reviews (EMSE 2023) • Kang et al., A Preliminary Evaluation of LLM-Based Fault Localization (MAPS 2023) • Hellendoorn and Sawant, The Growing Cost of Deep Learning for Source Code (CACM 2022)
Week 16. Final Project Presentation	
Final project report, video demo, and presentation slides due on 4/30 midnight	

Paper Reading Assignment

Starting from Week 4, you should expect to read two research papers on a specific topic in AI-assisted Software Engineering for each class. For each paper, you need to submit a short paper review (one or two paragraphs) in the form of questions and comments, no matter you are the presenter or not. Before each class, the instructor will create a post for each paper on Ed Discussion to collect students' reviews. You should post your review by replying to that post before the class (12pm that day). After posting your review, please also read other students' reviews to check whether others raised similar comments or questions like you or whether others raised comments or questions you have not thought of. During the paper discussion in class, please feel free to share comments you are super excited about or disagree and discuss with other students.

The grading of your paper review will depend on the overall quantity and quality of your questions and comments. As you read a paper or write your review, focus on the following perspectives.

- **Motivation of the work.** If the paper presents a new tool, who are the target users? Do they really need such a tool? What pain points does this tool address for those users? If the paper presents an empirical study, what are the research questions this study aims to answer? How important are these studies? Who will care about the findings and why should they care?
- **Novelty and significance of the work.** What is new here? What are the main contributions of the paper? What did you find most interesting?
- **Limitations, flaws, and blind spots.** Are there any unrealistic or false assumptions about the target users or the approach? Are there flaws or mistakes in the tool design, technical approach, or the study design?
- **Future work.** How would you improve on this work? Does this paper inspire any new ideas in your own research?

Paper Presentation and Discussion

Each student will present one research paper during the semester. The instructor will ask students to sign up papers to present by the end of the second week. The instructor will present the rest of the unselected papers during the semester.

Each paper presentation should be no more than 30 minutes, so we can have enough time for discussion. The presentation should focus on elaborating the motivation, related work, tool/study design, research questions, findings, limitations, and future work of the assigned paper. To make your presentation more insightful, try to center your presentation based on the literature and tell the audience why this work is proposed in the first place, how it advances people's understanding about a topic, and how it is different from other related work in the past. You are also encouraged to connect the

assigned paper to your own research. You should prepare for a set of questions (either came up by yourself or based on questions other students post on Ed Discussion) and co-lead an in-class discussion with the instructor based on these questions after the presentation.

The in-class discussion will follow the think-pair-share format.
- 1) Think. The presenter or the instructor will provoke students' thinking with a question. The students should take one or two minutes just to THINK about the question.
- 2) Pair. Using designated partners (such as with Clock Buddies), nearby neighbors, or a deskmate, students PAIR up to talk about their answers with each other. They compare their mental or written notes and identify the answers they think are best, most convincing, or most unique.
- 3) Share. After students talk in pairs for a few minutes, the presenter or instructor will call for pairs to SHARE their thinking with the rest of the class.

Course Project Instructions

You are expected to work on a course project either alone or in groups (3 to 5 students in a group). You can pick any topics related to AI-based software engineering, e.g., a new LLM-based code generation pipeline, a new code summarization model, a new ML-based bug detection tool, etc. At the end of Week 2, the instructor will release a list of sample project ideas with paper references, but feel free to work on your own ideas. Between Week 1 and Week 6, please stop by during office hours to discuss your project ideas with the instructor to get early feedback on the relevance, novelty, feasibility, and significance of your ideas.

A short project proposal is due on Feb 18 midnight (Week 6). This proposal should describe the project idea, the motivation of this idea, and (optional) a usage scenario if you propose to build a new tool. The proposal could be any length but no longer than 4 pages. It will be evaluated based on the quality of the idea and writing, not the length of your writing.

A mid-point project summary is due on Mar 24 midnight (Week 11). This summary should describe the envisioned approach/methodology/design as well as which parts have been done so far. The summary could be any length but no longer than 4 pages.

In Week 16, each team will deliver a presentation of their project. The presentation will be about 20 minutes. You will get another 5 minutes for Q&A.

You should also prepare a 5-min video demonstration of your work. If you develop a tool, your video should focus on demonstrating the features or functionality of your tool. If you

conduct an empirical study, your video should focus on demonstrating the key findings and presenting examples to elaborate the findings.

Your presentation slides and video demo need to be submitted on Brightspace by Apr 30 midnight.

The final project report is due on Apr 30 midnight. Your project report should be in the double-column ACM conference format. The instructor will provide both a word and latex template. Your final project report should not exceed 10 pages, not including references and appendices. Feel free to reuse sections from your midpoint project report in your final report.

Your final report should be structured like a conference paper. It should contain:

- Abstract
- A well-motivated introduction
- Related work with proper citations
- Description of your methodology
- Evaluation results
- Discussion of your approach, threats to validity, and additional experiments
- Conclusions and future work

If you are doing a project that involves implementation, please include a link to your Github repository in your final report. Please also add a README file in your repository to describe how to run and test your code.

Important dates:

- Project proposal due on 2/18 midnight
- Mid-point project summary due on 3/24 midnight
- Final project presentation on Week 16
- Final project report, presentation slides, and video demo due on 4/30 midnight

Quizzes

We will have five pop quizzes during this course. Each quiz should take about 5 to 10 minutes. Each quiz will assess your understanding about the research topics covered in the previous two or three weeks. The quizzes will include True or False questions, multiple-choice questions, and/or open-ended questions. To prepare for the quizzes, make sure (1) you have read the papers, (2) review the slides from the instructor or the paper presenter, (3) understand the methodologies, findings, and contributions of each paper.

The instructor cannot accommodate quizzes on a different date, except for cases of grief/bereavement, medical reasons (doctor's note required), military service, jury duty, and parenting leave. For these exceptional cases, the student or the student's representative should contact or go to the Office of the Dean of Students website to complete appropriate forms for instructor notification. For details, see the Academic Regulations & Student Conduct section of the University Catalog website.

Course Policies and Expectations

Late submissions

Late submissions are accepted with 20% decaying credit per day.

Attendance

This course follows Purdue's academic regulations regarding attendance, which states that students are expected to be present for every meeting of the classes in which they are enrolled. Your final grade will depend on your participation in the class. Please come to the class continuously, read the assigned papers, and participate in discussions.

If you feel sick, have any symptoms associated with COVID-19, or suspect you have been exposed to the virus, you should stay home and contact the Protect Purdue Health Center. Please also notify the instructor so that the instructor can arrange remote participation for you. If you miss classes because of COVID-related reasons, your final grade will not be affected by your absence of classes. For more guidance on class attendance related to COVID-19 are outlined in the Protect Purdue Pledge for Fall 2021 on the Protect Purdue website.

For other conflicts or absences, such as for many University-sponsored activities and religious observations, the student should inform the instructor of the situation as far in advance as possible. When the student is unable to make direct contact with the instructor and is unable to leave word with the instructor's department because of circumstances beyond the student's control, and in cases falling under excused absence regulations, the student or the student's representative should contact or go to the Office of the Dean of Students website to complete appropriate forms for instructor notification. Under academic regulations, excused absences may be granted for cases of medical reasons (doctor's note required), grief/bereavement, military service, jury duty, and parenting leave. For details, see the Academic Regulations & Student Conduct section of the University Catalog website.

Grief Absence Policy for Students

Purdue University recognizes that a time of bereavement is very difficult for a student. The University therefore provides the following rights to students facing the loss of a family member through the Grief Absence Policy for Students (GAPS). GAPS Policy: Students will be excused for funeral leave and given the opportunity to earn equivalent credit and to demonstrate evidence of meeting the learning outcomes for misses assignments or assessments in the event of the death of a member of the student's family.

Intellectual Honesty

Please read the departmental academic integrity policy. This will be followed unless we provide written documentation of exceptions. You should also be familiar with the Purdue University Code of Honor and Academic Integrity Guide for Students. You may also find Professor Spafford's course policy useful - while we do not apply it verbatim, it contains detail and some good examples that may help to clarify the policies above and those mentioned below.

In particular, we encourage interaction: you should feel free to discuss the course with other students. However, unless otherwise noted work turned in should reflect your own efforts and knowledge.

For example, if you are discussing an assignment with another student, and you feel you know the material better than the other student, think of yourself as a teacher. Your goal is to make sure that after your discussion, the student is capable of doing similar work independently; their turned-in assignment should reflect this capability. If you need to work through details, try to work on a related, but different, problem.

If you feel you may have overstepped these bounds, or are not sure, please come talk to us and/or note on what you turn in that it represents collaborative effort (the same holds for information obtained from other sources that provided substantial portions of the solution.) If we feel you have gone beyond acceptable limits, we will let you know, and if necessary we will find an alternative way of ensuring you know the material. Help you receive in such a borderline case, if cited and not part of a pattern of egregious behavior, is not in our opinion academic dishonesty, and will at most result in a requirement that you demonstrate your knowledge in some alternate manner.

Use of Copyrighted Materials

Students are expected, within the context of the Regulations Governing Student Conduct and other applicable University policies, to act responsibly and ethically by applying the appropriate exception under the Copyright Act to the use of copyrighted works in their

activities and studies. The University does not assume legal responsibility for violations of copyright law by students who are not employees of the University.

A Copyrightable Work created by any person subject to this policy primarily to express and preserve scholarship as evidence of academic advancement or academic accomplishment. Such works may include, but are not limited to, scholarly publications, journal articles, research bulletins, monographs, books, plays, poems, musical compositions and other works of artistic imagination, and works of students created in the course of their education, such as exams, projects, theses or dissertations, papers and articles.

Violent Behavior Policy

Purdue University is committed to providing a safe and secure campus environment for members of the university community. Purdue strives to create an educational environment for students and a work environment for employees that promote educational and career goals. Violent Behavior impedes such goals. Therefore, Violent Behavior is prohibited in or on any University Facility or while participating in any university activity.

Emergencies

In the event of a major campus emergency, course requirements, deadlines and grading percentages are subject to changes that may be necessitated by a revised semester calendar or other circumstances beyond the instructor's control. Relevant changes to this course will be posted onto the course website or can be obtained by contacting the instructors or TAs via email or phone. You are expected to read your @purdue.edu email on a frequent basis.

Accessibility and Accommodations

Purdue University strives to make learning experiences as accessible as possible. If you anticipate or experience physical or academic barriers based on disability, you are welcome to let me know so that we can discuss options. You are also encouraged to contact the Disability Resource Center at: drc@purdue.edu or by phone: 765-494-1247.

Nondiscrimination

Purdue University is committed to maintaining a community which recognizes and values the inherent worth and dignity of every person; fosters tolerance, sensitivity, understanding, and mutual respect among its members; and encourages each individual to strive to reach his or her own potential. In pursuit of its goal of academic excellence, the University seeks to develop and nurture diversity. The University believes that

diversity among its many members strengthens the institution, stimulates creativity, promotes the exchange of ideas, and enriches campus life.

Purdue University prohibits discrimination against any member of the University community on the basis of race, religion, color, sex, age, national origin or ancestry, genetic information, marital status, parental status, sexual orientation, gender identity and expression, disability, or status as a veteran. The University will conduct its programs, services and activities consistent with applicable federal, state and local laws, regulations and orders and in conformance with the procedures and limitations as set forth in Executive Memorandum No. D-1, which provides specific contractual rights and remedies. Any student who believes they have been discriminated against may visit University's website (www.purdue.edu/report-hate) to submit a complaint to the Office of Institutional Equity. Information may be reported anonymously.