https://doi.org/10.2352/EI.2026.38.3.MOBMU-317

Development of a Mobile Application for the Community-Focused Microclimate-Informed Indoor Heat Emergency Alert (CommHEAT) System

Alex Raymond Renner; VRAC Research Center, Iowa State University; Ames, Iowa/USA
Samantha Edwards; VRAC Research Center, Iowa State University; Ames, Iowa/USA
Joel McCleary; VRAC Research Center, Iowa State University; Ames, Iowa/USA
Adam Kohl; VRAC Research Center, Iowa State University; Ames, Iowa/USA
Kexin Wang; VRAC Research Center, Iowa State University; Ames, Iowa/USA
Eliot Winer; VRAC Research Center, Iowa State University; Ames, Iowa/USA

Abstract

The mobile application for the Community-Focused Microclimate-Informed Indoor Heat Emergency Alert (CommHEAT) system forecasts indoor temperatures and heat indices for the next seven days for users and their "community" friends. The primary challenge in the mobile application development was creating a system that connected external weather data sources, an indoor temperature simulation (EnergyPlus) that requires High Performance Computing (HPC), and a web-based database. Python scripts were used to retrieve input data from external weather websites (e.g., Mesonet and the National Weather Service) and empirical data for dwelling "archetypes" from spreadsheets, run simulations, and export the results to a MySQL database created for the system. The mobile application was created with Unity and forecast information was extracted from the database using C# and web-based PHP scripts for security.

Interdisciplinary rhetoric challenges and mid-development requests for additional application features caused numerous modifications and replacements of methods to create the required mobile application features. Despite these challenges, the development team successfully connected each component of the CommHEAT system and delivered a fully functional application that was deployed to iOS and Android mobile phones for a user study conducted in the summer of 2025.

Introduction

Academic research teams should collaborate with experienced developers early on in research projects to ensure that systems are in place to support development of the mobile application. This paper presents challenges that were faced when a research team at Iowa State University (ISU) collaborated with experienced developers near the end of a three-year NSF funded research project [1]. The Community-Focused Microclimate-Informed Indoor Heat Emergency Alert (CommHEAT) system to delivers personalized indoor temperature and heat index predictions to residents using a mobile application. The CommHEAT research team began collaborating with experienced developers and database administrators at the VRAC research center at ISU in April 2025. The mobile application needed to be developed in less than two months for an Institutional Review Board (IRB) user study that would be conducted with participants during the summer of 2025. In this research work, "VRAC developers" refers to the experienced developers and database administrators consisting of students, staff, and a faculty member at the VRAC research center. "CommHEAT team" refers to the interdisciplinary team that had been working on the research project. The CommHEAT team created "archetypes" attributes of dwellings that would affect the indoor temperatures, heat transfer models for a building energy simulation, and a UI design for the mobile application. However, they had not created a system to collect and combine data needed to run the energy simulation and make predicted indoor temperatures available for display in the mobile application.

The primary technical challenge was building a system with a complete data pipeline that could store, send, retrieve, and parse data from multiple external sources before indoor temperature and heat index forecasts could be displayed to the user. When a user registered, personal information had to be securely stored on Android or iOS devices and sent to a central database. During the first login, the app verified credentials against the database and stored them locally on their mobile device. The application needed to retrieve and calculate minimum and maximum values across all of the user's archetypes with and without Air Conditioning (AC) for the current and next-12-hour periods and repeat this process for each of the user's registered friends. Offloading calculations to server-side PHP scripts that generated JSON files storing indoor temperature and general user data enabled secure, efficient parsing in Unity C# scripts. An efficient local data structure needed to be created to provide a responsive User Interface (UI) that updated indoor temperatures following UI interaction.

This complex system that was developed in under two months (late April to early summer 2025) required the creation of a data pipeline and database to store data. Data was prepared so that an indoor temperature simulation could generate predicted indoor temperatures for the next 7 days. Once the weather prediction data was generated and stored in the centralized database, the system delivered real-time indoor weather forecasts via a cross-platform mobile application. The overall system, illustrated in Figure 1, depicts the CommHEAT system architecture, which is comprised of numerous components and complex connections. Arrows represent the sequential data flow: external weather APIs feeding into Python preprocessing scripts, archetype data merging, EPW file generation, parallel EnergyPlus jobs on the HPC cluster, automated upload to MySQL tables, PHP-mediated JSON retrieval, and lastly rendering in the Unity mobile application on user devices.

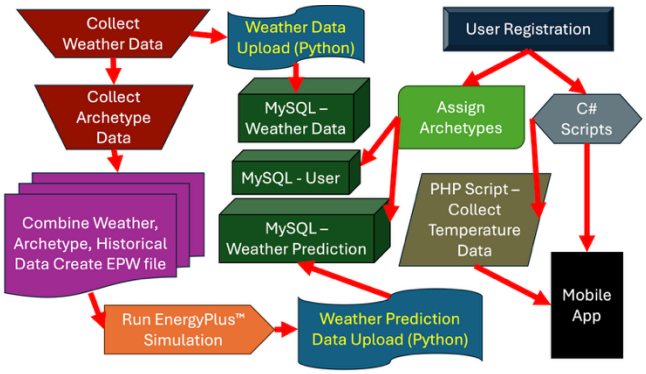
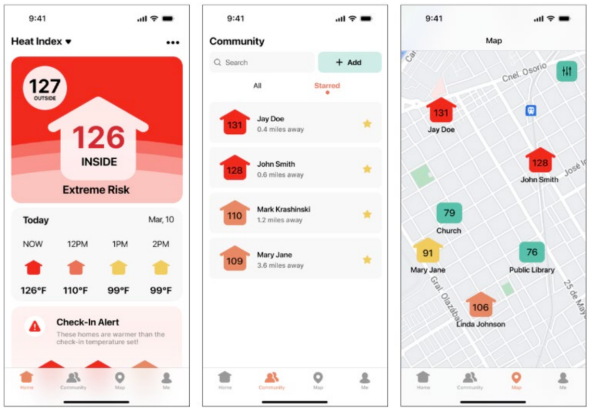


Figure 1: CommHEAT system and application architecture.



(a) (b) (c)

Figure 2: UI design screens (a) main temperature display, (b) community monitoring, (c) map and cooling-center locator.

Prior Work

Prior CommHEAT research demonstrated that indoor dwelling temperatures vary significantly across dwelling types, often exceeding outdoor "feels-like" temperatures and creating risks during extreme heat events [3]. Extreme heat poses significant health risks, particularly for vulnerable populations in urban environments. The CommHEAT research team had been developing energy models to predict indoor temperatures and heat indices for over two years, with an interdisciplinary team of architects, mechanical engineers, and Human Computer Interaction (HCI) experts. Empirical data for "archetypes" were generated by architects in the research team to account for variables such as building materials, insulation, shade, window size and location, dwelling orientation, and occupancy for dwellings in the test area of Des Moines, Iowa. Archetypes and weather data helped mechanical engineers develop heat transfer models for the EnergyPlus building energy simulation program [8].

HCI experts developed the design for a User Interface (UI) to be used in a mobile application informing users of the temperature and heat indices of their dwelling with and without AC. A user study evaluated perceptions and usability of the CommHEAT UI, confirming that residents needed alerts that combined air temperature, heat index, AC status, and community connectivity [2, 3]. The UI design for the mobile application includes three screens, shown in Figure 2. Figure 2a shows the main temperature display featuring large numerical readouts for current indoor air temperature or heat index, color-coded risk bands and toggle buttons for AC status. Figure 2b shows the community monitoring screen, with a scrollable list of connected friends with thumbnail icons, current average indoor temperatures, and alert icons. Figure 2c shows the integrated map and cooling-center locator to pinpoint nearby public cooling facilities.

Methodology

Overview

The VRAC developers chose to create the mobile application in Unity because it provides a robust cross-platform build system, and extensive support for UI customization. The overall methodology involved constructing an end-to-end data pipeline that integrated real-time and historical weather data from external APIs, empirical archetype information from spreadsheets, EnergyPlus simulations executed on an HPC cluster, automated storage in a MySQL database, and secure retrieval via PHP for display in the Unity-based mobile app. Creating the system with its data pipeline to the MySQL database, enabled VRAC developers to efficiently retrieve data for display in the CommHEAT mobile application.

External Data Sources

The data integration pipeline began with custom Python scripts written to collect weather data from external sources such as the Iowa Environmental Mesonet [7] and the National Weather Service [6]. Numerous challenges arose from the unique APIs of these websites, including inconsistent data formats, mixed units (e.g., Celsius/Fahrenheit), and irregular update schedules. These were overcome through rigorous error logging, custom validation functions, and the use of Python libraries such as pandas and datetime. Historical weather data from established typical-year spreadsheets were merged with current forecasts enhancing prediction accuracy.

EnergyPlus Simulation

The combined weather data were then integrated with empirical user-specific archetypes (building materials, shade levels, window orientation, insulation R-values, etc.) to generate EPW input files for EnergyPlus [8]. Close collaboration with the CommHEAT team was essential, as earlier scripts could not be reused for EPW file generation. Since each user could have up to three different archetypes it was essential to maintain explicit file naming conventions for Python scripts to generate all EPW files. Strict EPW formatting requirements necessitated detailed logging to

prevent simulation failures that would waste HPC resources and produce incorrect indoor temperature predictions. Once EnergyPlus simulations completed on the ISU HPC cluster, Python scripts extracted hourly indoor air temperatures, heat indices and AC conditions for the next seven days for every user archetype and automatically uploaded the data to MySQL database tables [9].

UI Implementation

The original UI design shown in Figure 2 was created with Figma [4]. Converting the Figma-designed UI into Unity uGUI [5] presented significant challenges. Automated import tools produced unsatisfactory results, requiring manual recreation of every screen in Unity. Since usability evaluations had already validated the UI [2], replication of colors, button shapes, sizes, and layout was mandatory. A dedicated Unity scene was created for each application page, utilizing canvas scaling for device independence. Figure 3a shows the main temperature display with large dynamic readouts, risk-color backgrounds, and toggle controls. Figure 3b shows the community screen with temperature badges and alert icons for user's friends. The map and cooling-center locator is shown in Figure 3c. The application concept screen shown in Figure 3d explains the microclimate modeling. The symptoms and first aid screen is shown in Figure 3e. Figure 3f shows the user profile screen with information collected during registration.

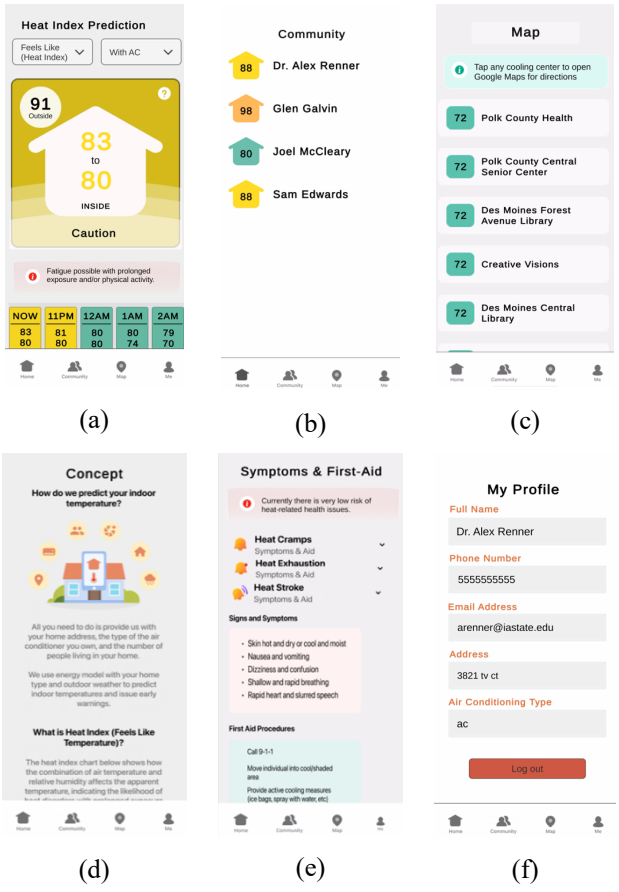


Figure 3: Unity UI screens for (a) Main temperature display, (b) Community monitoring, (c) Map and cooling center-locator, (d) Application Concept, (e) Symptoms and First Aid, (f) User Profile.

Registration and Login

Deep linking to the registration URL enabled a seamless return-to-app button in the device's default browser. Upon registration, the system automatically notified CommHEAT team members via email so that dwelling archetypes could be promptly assigned. User credentials were securely stored locally after the first login using Unity PlayerPrefs for convenient future access. A secure web interface hosted on the VRAC server (compliant with IRB protocols) eliminated the need for team members to use PHPMyAdmin directly. Figure 4a shows the registration webpage with required fields for email, password, and user information (e.g., dwelling address). Figure 4b shows the login screen that users would only need to use once unless they logged out of the application.

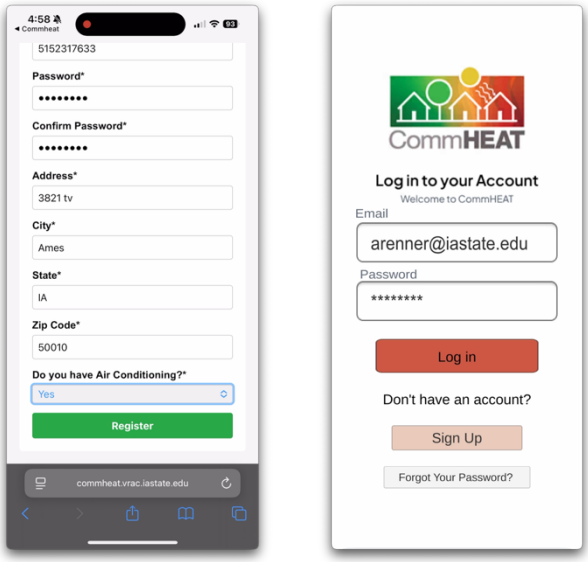
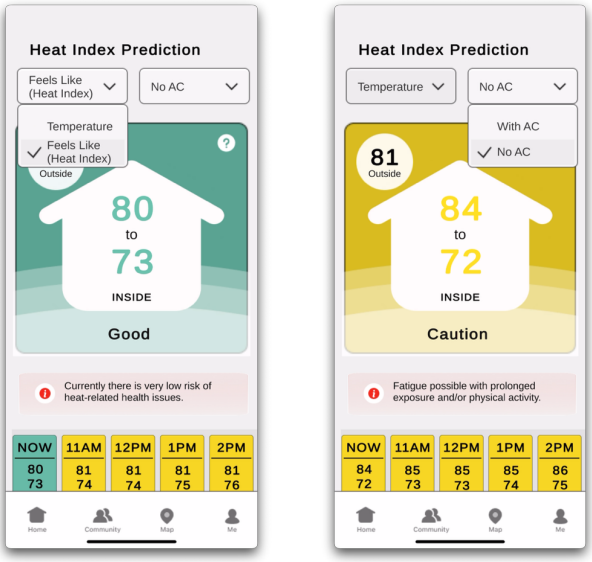


Figure 4: (a) User registration webpage, (b) Login screen.

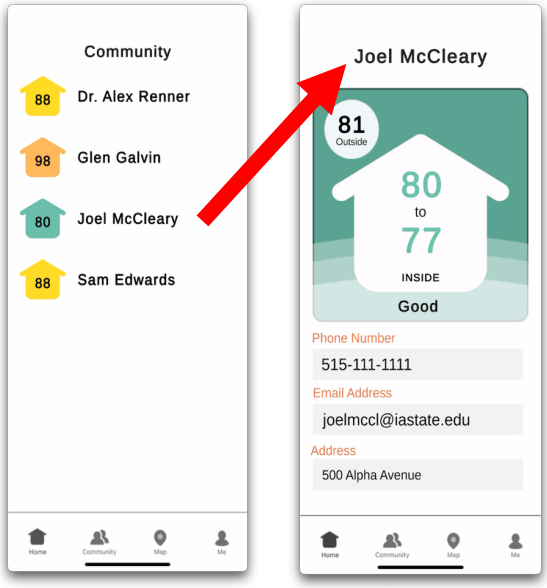
UI Data Retrieval

All hourly temperature values for the next seven days are loaded onto the mobile device when the application is opened. When users toggle between air temperature or heat index and with or without AC conditions, as shown in Figure 5, the current and next-12-hour minimum and maximum values update dynamically. A single Unity coroutine calls one PHP script that performs all necessary operations: querying user temperature values, calculating min/max values across all archetypes for air temperature and heat index (with/without AC), retrieving the user's friends, repeating the calculations for each friend, and returning the complete dataset as a JSON file using JSON_PRETTY_PRINT. UnityWebRequest POST methods with WWWForm handled communication with the PHP script on the same server as the MySQL database. Coroutines ensured data integrity regardless of connection quality. Intuitive toggles are shown in Figure 5a for the Temperature or Feels-Like (Heat Index) and in Figure 5b for the With AC or No AC conditions. When users toggle between these conditions, the displayed data values and risk colors instantly refresh using pre-loaded local data structs.



(a) (b)

Figure 5: (a) Temperature or Feels Like (Heat Index) toggle, (b) With AC or No AC toggle.



(a) (b)

Figure 6: (a) Community friends overview, (b) Friend detail view.

Local Data Structs

To aid the application's speed on a mobile device, a schema to store, access, and request essential data was developed. First, a public static class, *DataWarehouse*, was constructed to store the user's and associated friends' temperature data for project-wide access while using the application. This class enabled quick data access for information essential to populate the UI. Obtaining the temperature data required a controller class responsible for querying a server via PHP protocols. This controller class was able to parse the queried JSON data in a format that could be safely stored in the *DataWarehouse*. The data was formatted and populated according to a series of class structs. These structs not only allowed for the data to be populated and stored in *DataWarehouse* but also provided safe methods to extract relevant data upon request.

User Data Manager

A public static singleton User Data Manager handles all UI events, login/profile data, and extraction of temperature information from the Data Warehouse. It populates the community page with friends' names and current average indoor temperatures and opens detailed sub-pages showing worst-case heat-index values and contact information. Figure 6a illustrates the community friends overview screen with cards displaying each friend's name and current average indoor temperature. Figure 6b shows the friend detail view screen enabling users to check on the status of their friends and obtain their contact information. The singleton pattern prevented data duplication and ensured thread-safe updates across scenes.

Results and Discussion

Server-side PHP Data Query

Multiple coroutines slowed the MySQL server and increased mobile processing load. Every time the temperatures values needed to update, a sequence of coroutines would need to retrieve temperature data, then do processing on the mobile device which would be too slow for user study purposes. Consolidating all operations into a single PHP script proved most efficient, enabling fast parsing and instantaneous UI updates after the initial 20–60-second data load. While this load time is a minor inconvenience, it only occurs once when the user starts the application. If Unity C# coroutines were used, every time a user toggled between air temperature and heat index or with AC or not, the application would have taken at least 60 seconds.

Unity MySQL Queries

Direct MySQL access from C# scripts worked in the Unity editor simulator and on Android devices but failed on iOS devices because of firewall restrictions. The device simulator did not reveal the iOS firewall networking issue. The team therefore rebuilt the data-retrieval layer using PHP-mediated JSON within 3–4 days, ensuring full cross-platform compatibility. By addressing technical challenges in data integration, platform compatibility, and real-time UI updates, the development team created a robust mobile application that provides users with personalized heat risk information. The CommHEAT mobile application represents an innovative approach to community-focused heat emergency alerts, combining advanced microclimate simulations with a user-friendly, cross-platform mobile interface.

Conclusion

Python scripts efficiently collected forecasted weather data from external sources, merged it with historical typical-year data, and combined the result with dwelling archetypes to generate EPW files for EnergyPlus simulations on the HPC cluster. Python scripts were written to automate the process of populating the MySQL database with results from the EnergyPlus simulation. The data integration pipeline that generated EPW files from historical and forecasted weather data with dwelling archetypes was essential for the CommHEAT mobile application to retrieve and display indoor temperatures to users.

The shift from direct MySQL access to PHP-mediated queries resolved iOS security constraints and ensured cross-platform compatibility. PHP scripts were used to collect indoor temperatures as well as populate the mobile application with data specific to each user (e.g., their friends, user profile). PHP scripts used JSON encoding making it easier to parse the data. The responsive UI dynamically updates on user toggles with color-coded risk levels because data retrieved from the database was efficiently stored in local data structs providing instantaneous feedback after the initial load.

In summary, the successful delivery of a fully functional CommHEAT mobile application within an extremely compressed two-month timeline. The application was successfully deployed to both iOS and Android devices for the user study held in the summer of 2025. The resulting system not only meets the immediate needs of the NSF-funded user study but also establishes an architectural template for future community-focused environmental alert applications. This project highlights a critical lesson for academic research teams: engaging experienced developers at the earliest stages of application development design dramatically reduces technical challenges and accelerates translation of scientific models into real-world tools that protect vulnerable populations during extreme heat events.

References

[1] Award Abstract #2226880 SCC-IRG Track 2: A data-driven approach to designing a community-focused indoor heat emergency alert system for vulnerable residents (CommHEAT).

[2] T. Yao, M. C. Dorneich, R. Thomas, et al., “Evaluating User Perceptions and Usability of CommHEAT: A Community-Based Heat Alert Application,” Proc. Human Factors and Ergonomics Society Annual Meeting, vol. 69, no. 1, pp. 988-993, 2026.

[3] T. Yao, M. C. Dorneich, U. Passe, et al., “Vulnerable communities in the face of heat: a pilot study on perceptions, behaviors, and support networks during heat events,” Proc. Human Factors and Ergonomics Society Annual Meeting, vol. 68, no. 1, pp. 1890-1895, 2024.

[4] Figma documentation, https://www.figma.com.

[5] Unity uGUI Manual, https://docs.unity3d.com/Packages/com.unity.ugui@2.0/manual/index.html.

[6] National Weather Service, https://www.weather.gov.

[7] Iowa Environmental Mesonet, https://mesonet.agron.iastate.edu.

[8] EnergyPlus, https://energyplus.net.

[9] MySQL, https://www.mysql.com.

[10] PHP JSON functions, https://www.php.net/manual/en/json.constants.php.

Author Biography

Alex Raymond Renner is a Research Scientist 2 at the VRAC Research Center at Iowa State University, Ames, Iowa/USA. His work focuses on developing extended reality (XR) applications in a wide variety of research domains with over thirteen years of experience. He supports outreach activities for the VRAC Research Center performing demonstrations for visitors. He is a term faculty professor for the Mechanical Engineering department at ISU. He continues his Ph.D. research where he developed the only thermal process simulation application for 3D printing that allows users to explore tradeoffs between three print settings in real-time, reducing or eliminating part defects.

Samantha Edwards is a undergraduate student in Mechanical Engineering at ISU and works for the VRAC Research Center as an undergraduate research assistant.

Joel McCleary is a masters student in Mechanical Engineering at ISU and is a graduate research assistant at the VRAC Research Center.

Adam Kohl is a PhD student in Mechanical Engineering at ISU with a co-major in Computer Engineering and is a graduate research assistant at the VRAC Research Center.

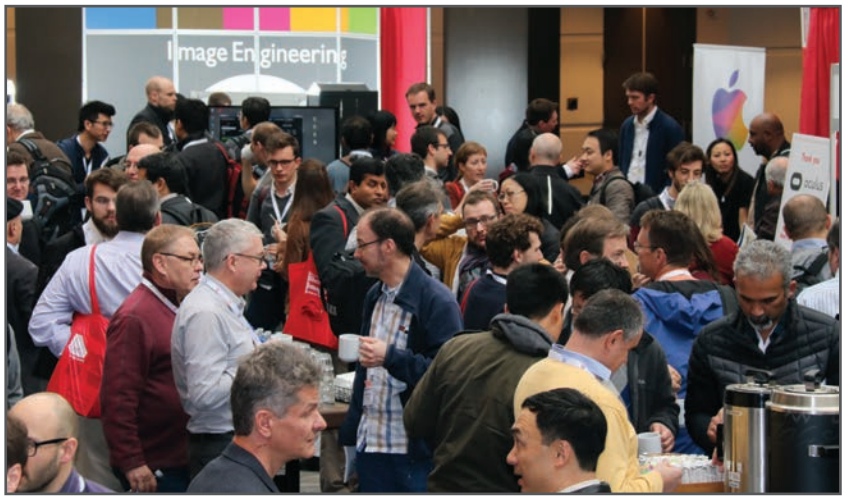
Kexin Wang is a PhD student in Human Computer Interaction at ISU with a co-major in Computer Engineering and is a graduate research assistant at the VRAC Research Center.

Eliot Winer is the director of the VRAC Research Center at ISU. He is a Professor in Mechanical Engineering with courtesy appointments in the departments of Electrical and Computer Engineering and Aerospace Engineering at ISU.

JOIN US AT THE NEXT EI!

electronic IMAGING

Imaging across applications . . . Where industry and academia meet!



• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

ADVANCING IMAGING SCIENCE 75 1947 IS&T 2022 imaging.org