JAMIA Open, 4(3), 2021, 1–7
doi: 10.1093/jamiaopen/ooab040
Application Notes

Application Notes

A simple electronic medical record system designed for research

Andrew J. King [1,2] Luca Calzoni [1] Mohammadamin Tajgardoon [3] Gregory F. Cooper [1,3] Gilles Clermont [2] Harry Hochheiser [1,3] and Shyam Visweswaran [1,3]

[1]Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, Pennsylvania, USA, [2]Department of Critical Care Medicine, University of Pittsburgh, Pittsburgh, Pennsylvania, USA and [3]Intelligent Systems Program, University of Pittsburgh, Pittsburgh, Pennsylvania, USA

Corresponding Author: Andrew J. King, 5607 Baum Blvd, Suite, 523, Pittsburgh, PA, 15206 USA (andrew.king@pitt.edu).

Received 8 November 2020; Revised 23 March 2021; Accepted 5 May 2021

ABSTRACT

With the extensive deployment of electronic medical record (EMR) systems, EMR usability remains a significant source of frustration to clinicians. There is a significant research need for software that emulates EMR systems and enables investigators to conduct laboratory-based human–computer interaction studies. We developed an open-source software package that implements the display functions of an EMR system. The user interface emphasizes the temporal display of vital signs, medication administrations, and laboratory test results. It is well suited to support research about clinician information-seeking behaviors and adaptive user interfaces in terms of measures that include task accuracy, time to completion, and cognitive load. The Simple EMR System is freely available to the research community and is on GitHub.

Key words: electronic health records, open-source software, user–computer interface, human–computer interaction, eye-tracking

BACKGROUND

The user interfaces of electronic medical record (EMR) systems are an assortment of legacy designs, billing requirements, and local customizations. The resulting misalignment of clinical and electronic workflow adds extraneous work and is a significant source of clinician frustration.[1,2] To remedy the ills of the EMR, greater attention needs to be placed on understanding EMR use, design consequences,[3] and potential improvements. Unfortunately, laboratory-based research of clinician-EMR interactions is hampered by the combination of software unavailability, lack of customizability, and nondisparagement agreements (gag clauses).[4] A need exists for an open-source EMR system designed for research.

Existing open-source EMR software, like Vista and OpenMRS,[5,6] are free and unrestricted by nondisparagement agreements but are challenging to customize because they implement a range of EMR core functions. Core functions—such as result and order management, communication, clinical decision and patient support, and administrative processes and reporting—increase the size and complexity of the software's codebase.[7]

Many of the core functions of full-fledged EMR systems are not essential to investigators researching specific types of EMR interactions. For example, an EMR system's result and order management views are sufficient to study patient data access patterns in the EMR (information-seeking behaviors),[8,9] user experience (cognitive load, fatigue, acceptance),[10,11] and task completion (time to task completion, the accuracy of task completion, unintended consequences).[3,12] A lean EMR system designed for rapid customization and use by investigators will fill a critical research need.

LAY SUMMARY

A significant barrier limiting human–computer interaction research on electronic medical record (EMR) systems is a lack of available software. Vendor-based EMR systems are often not available to faculty and—especially—student investigators. Existing open-source options (e.g., VistA, OpenMRS) may be challenging to deploy, customize the user interface, or insert deidentified/synthetic patient data. The Simple EMR System is an open-source Python package for visualizing EMR data in laboratory-based research studies. Its web-based user interface is rapidly deployable and readily customizable. By lowering the barrier for conducting EMR interaction studies, more investigators can work towards discovering methods for reducing the burdens of EMR use and improving the quality of patient care.

We report on the development, functionality, and implementation of a simple EMR visualization tool, useful for exploring the user interface designs and evaluations in laboratory settings. As an open-source tool built from widely used open-source components, this software package provides a starting point for several types of EMR studies and, with further enhancements, could be extended to satisfy additional types of research. By eliminating the software barrier of conducting EMR research, we hope that more investigators and students will work towards discovering ways of reducing the cognitive burdens and time requirements of EMR use.

OBJECTIVE

To accelerate human–computer interaction research focusing on EMR systems, we developed a Python package that emulates an EMR system's display functionalities. We describe its key features—including details of the user interface, how to add patient records, and its use in research—and suggest ways to contribute to this open-source package.

METHODS

We implemented an EMR system designed for displaying patient data in laboratory-based research studies. The EMR system, called Simple EMR System, is built in Python using Bitnami Django.[13] System components—including patient data, a user interface, and experimental business logic—follow a model view controller (MVC) architecture (see Figure 1).

MVC is a widely used software design pattern that provides a framework for implementing a client–server software application.[14] The *model* is an abstraction that allows data to be drawn from different data sources. To streamline adapting to different data sources, the Simple EMR System relies on loading scripts that extract source data for a specified number of patient encounters, transform the data into JavaScript Object Notation (JSON) data structures, and store the structures in JSON files ("demographics.json," "observations.json," "medications.json," and "note_panel_data.json"). When a patient's record is requested at runtime, these files are loaded from a subdirectory named after the patient's encounter id.

The *view* is the presentation layer and constitutes the information that is viewed by the user on the interface of the Simple EMR System. The interface is implemented using the Django template language, JavaScript, and various libraries (Bootstrap, jQuery, Highstock). A set of JSON files are used to configure which panel of the interface displays what data fields ("data_layout.json"); the display group, display name, default y-axis range, and default normal range for each observation field ("variable_details.json"); and, the medication route and display name for each medication type ("med_details.json"). Further customizations can be made by editing the Hypertext Markup Language template page, JavaScript, or Cascading Style Sheet files ("case_viewer.html," "emr_3.js," or "bs_3.css").

The *controller* manages the flow of information between the model and the view and the user's workflow. In the Simple EMR System, the controller is a combination of built-in Django functions and Python scripts implementing business logic for determining what content to serve to the client. By default, the workflow contains four steps: study selection, user selection, case selection, and case viewing. Studies are defined by creating a subdirectory in the project's "resources" directory and renaming it to a name that ends with the suffix "_study" (e.g., "SimpleEMRSystem\resources\demo_study\"). Within a study directory, users (i.e., participants) are defined in a JSON file ("user_details.json") along with the cases they are assigned, the cases they completed, and the date they last accessed the system. Cases assigned to a user must also be defined in a separate JSON file ("case_details.json"), where each case has an array of dictionaries that define how the case should be displayed during the study (including details such as the minimum and maximum time-to-display, and the instructions to display to the user). This data structure allows for complex workflows, for example, simulating morning rounds on subsequent days by including two dictionaries in the array: setting the first dictionary's maximum time-to-display to 8 AM during the first day of the patient encounter and setting the second dictionary's maximum time-to-display to 24 h later.

RESULTS

Simple EMR user interface

The Simple EMR System presents patient data temporally in four scrollable columns (see Figure 2). From left to right, the first panel contains vital signs, ventilator settings, and intake and output, the second panel contains medication administrations, the third panel contains laboratory test results, and the fourth panel contains clinical notes and reports. The user interface includes interactive features to make using the temporal representation tenable (see Figure 3). For example, users adjust the displayed time range by click-and-dragging start and endpoint markers on a timeline. The time adjustments update all the temporal data plots to keep all displayed data in sync. The plots have blue and peach regions to indicate the normal range (y-axis) and the most recent 24 h (x-axis), respectively. When an individual measurement is clicked, a vertical dotted line is placed at that time point on all plots. Further, the measurement value appears in a pop-up box when the cursor hovers over a data point.

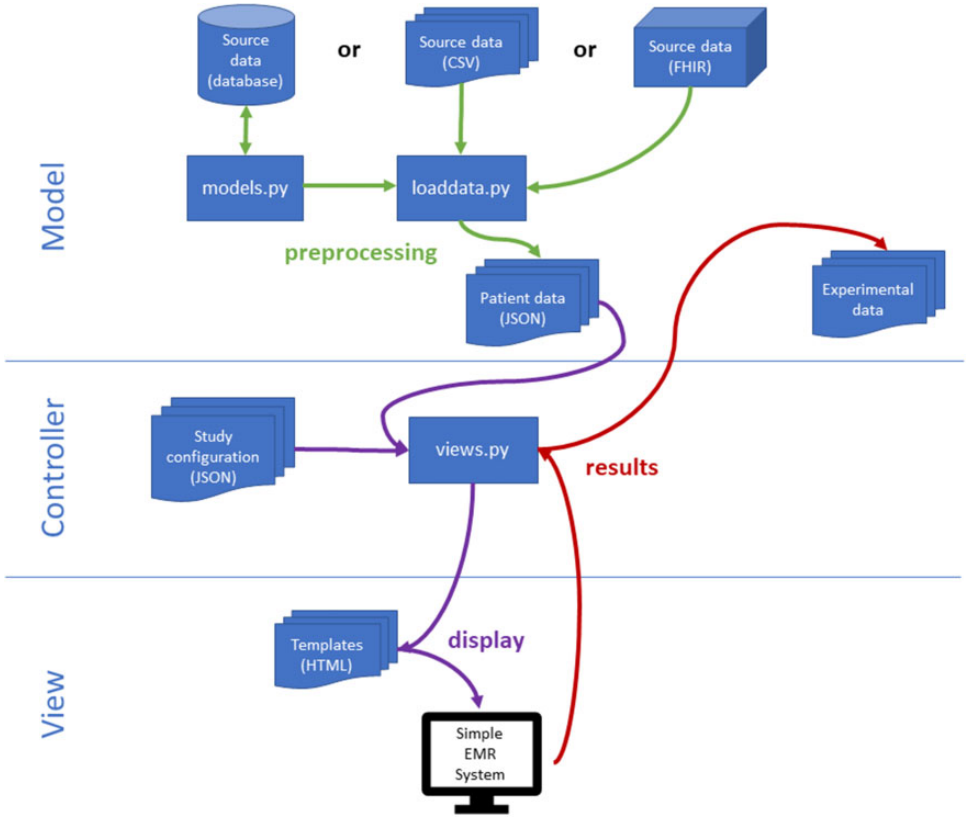

Figure 1. Simple EMR Systems' MVC architecture. Patient data are preprocessed to reduce load times. The case order and experimental conditions are managed in the controller via configuration files. The layout of the user interface is dictated by templates and accompanying JavaScript code. Experimental results, such as time to task completion or manual annotations of information-seeking behavior, are stored in text files.

The temporal design, layout, and interactive features were influenced by feedback from critical care physicians across multiple iterations of prototypes and research studies. The user interface has features that enable data collection for research. For example, study participants on reviewing patient cases can select relevant patient data by toggling a checkbox associated with a data element (see Figure 3). The lower right-hand region of the user interface provides instruction and navigation areas customized for the research study (see Figure 2). The system can be integrated with a low-cost eye-tracking device, as we have done, mounted at the bottom edge of the computer monitor.

The Simple EMR System software package with accompanying documentation is freely available on GitHub at https://github.com/ajk77/SimpleEMRSystem.

Patient cases

The Simple EMR System can be adapted to display patient data from any source. Its default support is for SyntheticMass, that is, Synthea CSV format.[15] To preprocess cases for display on the Simple EMR System, (1) download the source data and extract it into the "resources" folder. (2) Adjust parameters, such as the input and output directories, in the Synthea data loading script ("loaddata_synthea.py") and execute the script. (3) View details about each processed case in "list_case_dicts.json." (4) Select cases to use in the study and add them to "case_details.json" and (5) assign them to users in "user_details.json." Finally, (6) adjust display groups, medication routes, display names, and display locations in "variable_details.json," "med_details.json," and "data_layout.json."

The data loading script must be adapted when using data from other sources (e.g., MIMIC[16] or an FHIR server[17]). If connecting to an existing database, the Django Model API (application programming interface) can be used to streamline querying. Preprocessing all case data into JSON files is beneficial because it allows for faster server response times and because investigators can edit the processed JSON files directly.

At the time of publication, two example studies are included in the Simple EMR System repository: demo_study and synthea_study. Demo_study includes three synthetic patient cases created by scrambling the contents of deidentified records from a larger set of patients from a hospital's Cerner EMR system. Synthea_study includes 25 intensive care unit (ICU) encounters from the Synthea COVID-19 data set (available at https://synthea.mitre.org/downloads).[18]

Example research use

The Simple EMR System initially supported the Learning EMR project, where our goal was to build a data-driven method for prioritizing patient information in the EMR.[19,20] The project trained machine learning models to identify relevant patient data for physicians browsing the EMR before morning rounds.[21]

We used the Simple EMR System to collect clinician information-seeking behavior in two distinct ways: manually and with eye-tracking. In manual data collection, a checkbox appears next to the plot for each data field. The user toggles the checkbox to select/deselect data fields of interest, and the system stores the list of selected data fields in a text file.

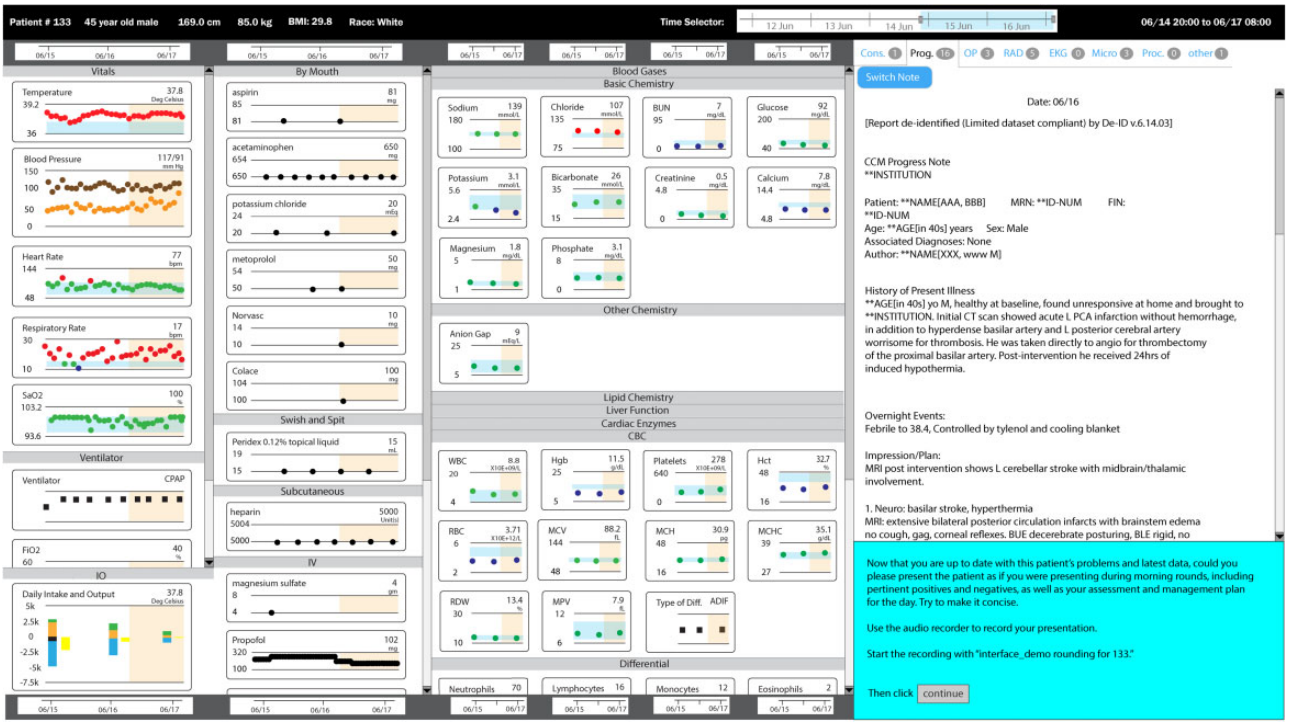

Figure 2. A screenshot of the user interface of the simple EMR system. Most patient data are shown using temporal plots. At the top in black, the interface shows patient demographics and a time selector, which is used for adjusting the display range for the temporal data plots. The data plots are organized into four scrollable columns. The left-most region shows vital signs, ventilator settings, and fluid intake and output. The next region shows medication orders organized by the medication route. The third region shows laboratory test results, organized by panels (e.g., basic chemistry). On the data plots, the range of the x-axis is determined by the blue-selected region of the time selector at the top of the window. The final, right-most region shows clinical notes and reports, organized by type (consultant, progress, operation, radiology, electrocardiogram, microbiology, procedures, and others). At the lower right-hand region is an instruction and navigation region that can be customized for research.

For data collection via eye-tracking, a low-cost eye tracker is mounted on the computer monitor's lower bezel. The Simple EMR System can be paired with a Tobii Eye Tracker 4C to capture both a stream of eye-gaze coordinates and a stream of position coordinates for what is displayed onscreen. An algorithm then automatically maps eye-gaze to onscreen data elements to generate a list of what the user was viewing.[22,23] The scripts and accompanying documentation for incorporating eye-tracking with the Simple EMR System are freely available on GitHub (https://github.com/ajk77/EyeBrowserPy).

As part of the same work, we also used the Simple EMR System to test different means of prioritizing patient information, including selectively highlighting relevant patient data by changing some data elements' background color.

DISCUSSION

We developed an open-source EMR system designed for research. The Simple EMR System fills a gap for investigators who need a readily customizable EMR interface. It is well suited to support research about clinician information-seeking behaviors and adaptive user interfaces in terms of measures that include task accuracy, time to completion, and cognitive load. Django's MVC framework provides several advantages to investigators. The view provided by the Django template language is easy to modify and quickly make changes to the elements displayed on the screen. The model can be bypassed if someone wishes to create patient cases in JSON format directly, rather than customizing the data loading script. Finally, the Python script and JSON files that control experimental parameters (e.g., users, case assignments, and case workflow) can also be modified independently from either the view or model.

The Simple EMR System is limited to the display functions of an EMR system. Although this limitation precludes it from being used in research involving order entry or other more complex EMR activities, the simple code base makes adoption and modifications easier. Compared to OpenMRS and VistA, the Simple EMR System is easier to deploy and customize for different experimental needs. Rapid and iterative deployment can keep research projects moving and enables A/B testing of many different interface layouts.

Investigators will find the Simple EMR System suitable where a compact view of a complex patient record, like an ICU stay, is useful. It is also particularly useful for displaying data such as physiological measurements, laboratory test values, and medication administrations in temporal plots. However, temporal plots may not always be suitable and some investigators may want such data to be displayed in tables as found in many commercial EMR systems. While not currently supported, tables can be easily implemented in HTML by editing the Django template ("case_viewer.html"). Currently, the Simple EMR System does not support the display of non-patient data such as electronic messages among clinicians.

The Simple EMR System does not interact with vendor EMR systems currently in use in healthcare settings. Our goal in developing this system was not to replicate an entire modern EMR but to offer a simple and useful system for displaying patient data that mimics a real EMR in important ways to support research in laboratory settings. However, the open-source implementation provides a path to-

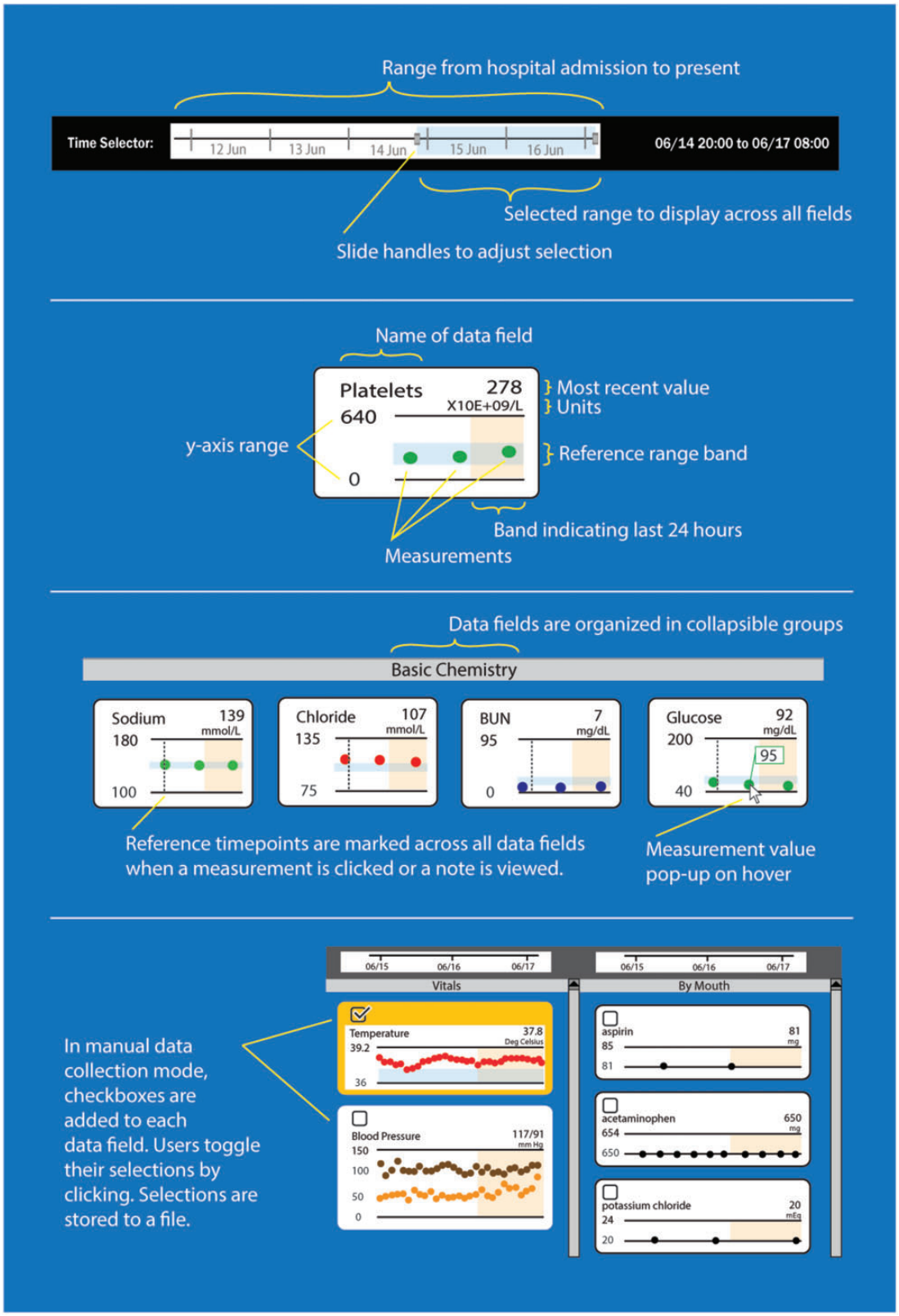

Figure 3. Details of some of the features of the user interface of the Simple EMR System. The first panel shows the adjustable time range that applies to all temporal plots. The second panel shows details of a single temporal data element, such as platelet count. The third panel shows that laboratory test results are organized into panels that are clinically meaningful, that reference timepoints can be added to aid users when interpreting data across multiple columns, and that individual measurements pop-up on cursor hover. The fourth panel shows data fields with checkboxes that are used to indicate selections.

Table 1. Community opportunities for contributing to the Simple EMR System

1. Include an out-of-the-box connection to alternative databases and data models (e.g., MIMIC-III, OMOP common data model)
2. Convert to a FHIR application.
3. Improve the stability of eye-tracking across environments.
4. Improve security so it can be used with live clinical data.
5. Convert to responsive HTML so that it renders attractively on different-sized devices.
6. Add interface layouts that mirror the "table of numbers" views of results and measurements that are the default view in commercial EMR systems.

Abbreviations: MIMIC-III: Medical information mart for intensive care III; OMOP: Observational medical outcomes partnership; FHIR: Fast Healthcare Interoperability Resources.

ward integration with a vendor system, as the Python scripts, we provide for loading of patient data might be modified to access a vendor's FHIR interfaces. Similarly, adapting the Simple EMR System to work within the SMART-on-FHIR framework is an exciting direction for future work.[24]

The Simple EMR System does not require input data to be mapped to any standardized terminologies. In the current implementation, problem lists, diagnoses, careplans, imaging studies, notes, and reports are displayed under different tabs of the note panel (right side of the screen). We welcome investigators to contribute to the Simple EMR System. It is available under the GNU General Public License v3.0. We list potential avenues for improvement in Table 1.

CONCLUSIONS

We developed a Simple EMR System and have made it available to the research community. It provides a platform for EMR human factors research that can save investigators time by allowing for custom EMR scenarios and interface layouts. Beyond accelerating our understanding of clinician perception and preferences when viewing patient data, the system may lead to new ways of presenting patient data that reduce the cognitive burden and alleviate some causes of burnout. We welcome contributors to this open-source project.

Contributors

All authors contributed to the design of the Simple EMR System. A.J.K. wrote all the software and led the authoring of the manuscript. All authors made critical revisions to the manuscript and approved the final version for submission.

ACKNOWLEDGMENTS

We thank the many critical care fellows who participated in the Learning EMR System studies and provided suggestions on how to improve the Simple EMR System, Brandon McLaughlin for scheduling rooms to conduct studies, and John Levander for introducing us to Bitnami Django Stack.

FUNDING

The research reported in this article was supported by the National Library of Medicine of the National Institutes of Health under award numbers T15 LM007059 and R01 LM012095. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Conflict of interest statement. None declared.

DATA AVAILABILITY

Data for interface demonstration is provided in the Simple EMR System GitHub repository at https://github.com/ajk77/SimpleEMRSystem/tree/master/resources.

REFERENCES

1. Kroth PJ, Morioka-Douglas N, Veres S, *et al*. Association of electronic health record design and use factors with clinician stress and burnout. *JAMA Netw Open* 2019; 2 (8): e199609.
2. Ruppel H, Bhardwaj A, Manickam RN, *et al*. Assessment of electronic health record search patterns and practices by practitioners in a large integrated health care system. *JAMA Netw Open* 2020; 3 (3): e200512.
3. Sittig DF, Wright A, Ash J, *et al*. New unintended adverse consequences of electronic health records. *Yearb Med Inform* 2016; 25 (1): 7–12.
4. Adler-Milstein J, Thao C. Why real-world health information technology performance transparency is challenging, even when everyone (claims to) want it. *J Am Med Informatics Assoc* 2020; 27 (9): 1462–5.
5. Brown S, VistA —U. S. Department of Veterans Affairs national-scale HIS. *Int J Med Inform* 2003; 69 (2–3): 135–56.
6. Mamlin BW, Biondich PG, Wolfe BA, *et al*. Cooking up an open source EMR for developing countries: OpenMRS - a recipe for successful collaboration. *AMIA Annu Symp Proc* 2006; 2006: 529–33.
7. Institute of Medicine Committee on Data Standards for Patient Safety. Key Capabilities of an Electronic Health Record System: Letter Report. In: Aspden P, Corrigan JM, Wolcott J, et al., eds. *Patient Safety: Achieving a New Standard for Care*. Washington, DC: National Academies Press; 2004: 430–70.
8. Kannampallil TG, Jones LK, Patel VL, *et al*. Comparing the information seeking strategies of residents, nurse practitioners, and physician assistants in critical care settings. *J Am Med Inform Assoc* 2014; 21 (e2): 249–56.
9. Kannampallil TG, Franklin A, Mishra R, *et al*. Understanding the nature of information seeking behavior in critical care: implications for the design of health information technology. *Artif Intell Med* 2013; 57 (1): 21–9.
10. Pollack AH, Pratt W. Association of health record visualizations with physicians' cognitive load when prioritizing hospitalized patients. *JAMA Netw Open* 2020; 3 (1): e1919301.
11. Khairat S, Coleman C, Ottmar P, *et al*. Association of electronic health record use with physician fatigue and efficiency. *JAMA Netw Open* 2020; 3 (6): e207385.
12. Kumar A, Aikens RC, Hom J, *et al*. *OrderRex clinical user testing: a randomized trial of recommender system* decision support on simulated cases. *J Am Med Inform Assoc*. 2020; 27 (12): 1850–9.
13. Django Software Foundation. Django [Internet]. 2013. https://www.djangoproject.com/ Accessed May 12, 2021.
14. Krasner GE, Pope ST. A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80. *J Object-Oriented Program* 1988; 1 (3): 26–49.
15. Walonoski J, Kramer M, Nichols J, *et al*. Synthea: an approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *J Am Med Informatics Assoc* 2018; 25 (3): 230–8.

16. Johnson AEW, Pollard TJ, Shen L, *et al.* MIMIC-III, a freely accessible critical care database. *Sci Data* 2016; 3 (2016): 160035.
17. HAPI FHIR [Internet]. http://hapi.fhir.org/ Accessed May 12, 2021.
18. Walonoski J, Klaus S, Granger E, *et al.* Synthea^{TM} Novel coronavirus (COVID-19) model and synthetic data set. *Intell Based Med* 2020; 1–2 (November 2020): 100007.
19. King AJ, Cooper GF, Clermont G, *et al.* Using machine learning to selectively highlight patient information. *J Biomed Inform 2019*; 100 (December 2019): 103327.
20. King AJ, Cooper GF, Hochheiser H, *et al.* Development and preliminary evaluation of a prototype of a learning electronic medical record system. *AMIA Annu Symp Proc* 2015; 2015: 1967–75.
21. King AJ, Cooper GF, Hochheiser H, *et al.* Using machine learning to predict the information seeking behavior of clinicians using an electronic medical record system. *Annu Symp Proc* 2018; 2018: 673–82.
22. King AJ, Hochheiser H, Visweswaran S, *et al.* Eye-tracking for clinical decision support: a method to capture automatically what physicians are viewing in the EMR. *AMIA Jt Summits Transl Sci Proc* 2017; 2017: 512–21.
23. King AJ, Cooper GF, Clermont G, *et al.* Leveraging eye tracking to prioritize relevant medical record data: comparative machine learning study. *J Med Internet Res* 2020; 22 (4): e15876.
24. Mandel JC, Kreda DA, Mandl KD, *et al.* SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. *J Am Med Informatics Assoc* 2016; 23 (5): 899–908.