COVID CV: A System for Creating Holistic Academic CVs during a Global Pandemic

Umesh Raja, Nahida Sultana Chowdhury, Rajeev R. Raje, Rachel Wheeler, Jane Williams, Aaron Ganci
Indiana University Purdue University Indianapolis
Indianapolis, Indiana, USA
{*uraja, nschowdh, rraje, wheelerr, jrwillim, aganci*}*@iupui.edu*

***Abstract*—The effects of the Covid pandemic have been, similar to the population at-large, unequal on academicians – some groups have been more susceptible than others. Traditional CVs are inadequate to highlight these imbalances. CovidCV is a framework for academicians that allows them to document their life in a holistic way during the pandemic. It creates a color-coded CV from the user's data entries documenting the work and home life and categorizing corresponding events as good or bad. It, thus, provides a visual representation of an academician's life during the current pandemic. The user can mark any event as major or minor indicating the impact of the event on their life. The CovidCV prototypical system is developed using a three tier architecture. The first tier, the front-end, is a user interface layer that is a web application. This prototype has a back-end layer consisting of two tiers which are responsible for handling the business logic and the data management respectively. The CovidCV system design is described in this paper. A preliminary experimentation with the prototype highlights the usefulness of CovidCV.**

***Keywords*-Client-Server architectural model; Object-relational mapping; Security; Web App.**

I. Introduction

The current global COVID-19 pandemic has devastated, similar to the population at-large, the academic community as well. In addition, its effects are felt differently by different populations in academia. Many studies have shown that women and people of color are bearing a disproportionate burden of service work [1] during the pandemic. Preliminary data also shows a significant decline of pre-prints and journal submissions by women [2] since the beginning of the pandemic. The fallout from the pandemic is expected to have a long-lasting effect on academic environments. Hence, there is a need for universities to devise alternative ways to address these inequalities. One way to tackle this challenge is to provide an innovative and adequate mechanism to the faculty to document a holistic picture of their life – a deviation from the traditional approach to create CVs, which only documents typical scholarly artifacts such as publications, grants, and presentations without providing the underlying "invisible context". Any alternate mechanism to document the academic life in current unpredictable times will not only assist the faculty and scholars to highlight this "invisible context" but also will provide a different perspective to the mentors and administrators in academia.

CovidCV is designed to address this challenge – it is a web-based system that will be beneficial to academicians to achieve their well-being and continuous advancement, and at the same, it can be used to foster an institutional change to achieve more diversity, equality, and inclusion. CovidCV provides a framework that: a) helps individual users with their professional development and well-being by documenting their work/life balance in a simple color-coded way; b) allows the creation of a comprehensive data-set that can form a basis for scholarly investigations related to gender and racial equity in the academy; and c) can be used as an effective tool by academic institutions to foster an inclusive environment. The CovidCV prototypical system (henceforth referred as the "CovidCV system") creates a color-coded CV from the user's data entries documenting work and home life, thus, providing in a simple, yet effective, visual representation of the real-life context of an academic life. CovidCV documents not only typical academic successes such as publications, promotions, grants, and awards, but also academic failures and struggles such as rejections, missed opportunities, instances of harassment, and disappointments. In addition, CovidCV allows recording of various aspects of home life such as important family events (e.g., birthday celebrations), life changing events (e.g., loss of a loved one), or ongoing struggles (e.g., caring for an elderly parent in the pandemic). The CovidCV system allows marking any event as Major or a minor event. CovidCV, hence, provides a comprehensive and holistic view of the conditions that significantly impact a faculty member's work life, thereby providing an appropriate context about the underlying invisible factors and make them visible.

In this paper, the design of the CovidCV system is discussed. The rest of the paper is organized as follows: Second section presents related work. Section III describes the CovidCV system architecture. Section IV explains the implementation details of the CovidCV system along with a discussion of a few experiments. The paper concludes by highlighting the insights gained and presents directions for future work.

This is the author's manuscript of the article published in final edited form as:

Raja, U., Chowdhury, N. S., Raje, R. R., Wheeler, R., Williams, J., & Ganci, A. (2021). COVID CV: A System for Creating Holistic Academic CVs during a Global Pandemic. 2021 IEEE International Conference on Electro Information Technology (EIT), 146–150. https://doi.org/10.1109/EIT51626.2021.9491917

II. RELATED WORK

University administrators across the country are struggling to develop appropriate responses to the impacts of Covid, from pausing tenure clocks to debating changes to review criteria to reflect the altered workload and increased burdens, to "reinvent[ing] what success in academia looks like" [3] [4]. They note that the pandemic is increasing the amount of time faculty will spend on teaching and service, leaving less time for scholarly endeavors. In addition, with women and minority faculty more likely to engage in mentoring and other service work, the impact is even more severe. Interestingly, a follow up letter to the paper noted above [3] insists that promotion and tenure committees need to carefully consider all activities faculty engage in during this time to promote equity. They recognize that the CV is the primary tool faculty use to communicate their work and that committees use to assess. They promote a COVID matrix as a mechanism to account for all the extra work and disruptions [5]. However, this matrix has been criticized because it does not necessarily reflect the inequitable outcomes that are more likely to occur to women and faculty of color [6]. The matrix does not comprehensively capture all aspects of faculty work and life. National foundations such as the American Council of Learned Societies [7] have shifted their funding priorities towards early-career scholars acknowledging the devastating impact, particularly on younger scholars who are more likely to have children at home. We believe that CovidCV addresses these shortfalls, as it captures the work and non-work changes that have impacted faculty life.

III. COVIDCV SYSTEM ARCHITECTURE

The CovidCV system, as shown in Figure 1, is designed as a web-based client-server application and has a 3-tier architecture. The front-end is a GUI-based tier that is separated from the back end layer containing business logic and the database tiers. The advantages of this separation are: a) all tiers can evolve independently, b) the front-end carries out minimal data management resulting in a thin client and thus, achieves quick response, and c) additional front-end layers, such as a mobile interface, when created will be able to utilize a single back-end. Below each of these layers are discussed.

A. Front-end Layer

The front-end layer or the user interface layer is built using JavaScript framework called *React.js* [8]. The framework considers a website as single page application – i.e., it instead of rending the entire web-page in the event of any changes renders only the modified content based on the user input. Such rendering improves the response time and provides a better user experience – both features that are critical for an interactive application such as CovidCV. The elements displayed on interactive pages are built using

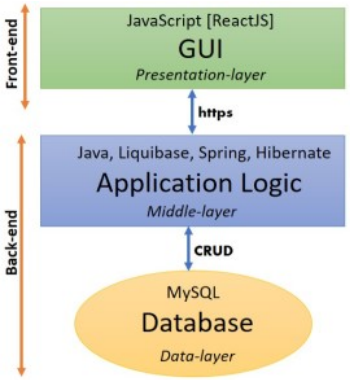

Figure 1: CovidCV System Architecture.

HTML and are styled using CSS. The React.js framework is built on top of the *Node.js* [9] framework that contains the *Node Package manager* (NPM). The NPM helps to resolve external dependencies and provides necessary tools and servers to run node applications. Therefore, the front-end is deployed using the NPM and we have used the *nginx* server [10] to deploy the CovidCV system.

B. Back-end Layer

The application logic layer is programmed in Java-8.0 using the *Spring boot* framework [11]. The Spring boot framework provides powerful methods to build *RESTful* APIs that enable clients to execute the code from different endpoints. This framework has a module called Spring Security [12] that allows many different login methods depending on application domain. The Spring boot framework, in addition, supports Aspect oriented programming to separate cross-cutting concerns such as login and transaction management from core application logic. The application logic layer utilizes *Hibernate* framework that is built on top of *Java Persistence API* (JPA) [13]. The purpose of *Hibernate* is to provide object-relational mapping to map Java objects to tables in a relational database. The *Liquibase* [14] is used to set up the database before running the application logic code. In order to satisfy the external dependencies, *Apache Maven* [15] is also used in this layer. The application logic layer requires a web container to accept https requests from clients and to transfer them to execute appropriate endpoints – *Tomcat* [14] server is used in the application logic layer as the web container.

C. Database Layer

The database layer is the third tier of the CovidCV system where the data actually resides. *MySQL* [16] database is used in the CovidCV prototype because it is open source. The only action that is needed in this tier for the setup process is to create a database named *covidcv*. The back-end code then automatically creates all the necessary data tables within the *covidcv* database and pre-populates it appropriately.

D. Communication

Currently, all three tiers are deployed on virtual machines running on different ports. The front-end layer communicates with the back-end layer by calling APIs over the secure HTTP (*https*) protocol. Using https, the data is encrypted before sending it over the network. Hence, the requests sent to the server are protected from eavesdropping or man in the middle attacks. The application logic tier communicates with the database tier for necessary Create, Read, Update or Delete (CRUD) operations.

E. Security

Since CovidCV documents many specific personal details of users, the privacy and security of that data is an inherent requirement of the CovidCV system. As mentioned earlier, the communication from the front-end layer to the application logic layer is carried over https, so that the data sent over the network is encrypted. In addition, critical data items of users such as name, email, and password are stored in the database in an encrypted format. The encryption of name and email is being achieved using JPA's built-in support that is based on *Advanced Encryption Standard (AES)* [17]. The password is encrypted using the *bcrypt* cipher that even developers themselves cannot decrypt. Spring security provides *PasswordEncoder* class that achieve this *bcrypt* encryption. However, to change a password, an user's current password needs to be properly matched. *PasswordEncoder* at that point only returns true/false if the user's input password matches the password stored in the database or not.

IV. CovidCV System Implementation

A. Front-end Design

As users of the CovidCV system are expected to be academicians from various fields, they may not be computer savvy, the user interface of the CovidCV system had to be friendly, yet effective. The design of the user interface was developed using a user-centered, iterative process. We first identified the type of users that we were targeting with this application: underrepresented academic faculty in the U.S. who are already overloaded with tasks at home and work. With this in mind, we designed the application with a priority on ease of use and familiarity. We used a principle in Human-Computer Interaction called "Jakob's Law" that puts forth the notion that users spend most of their time on sites other than your own and that, to work well, they should act in ways they are already accustomed [18]. To that end, we followed many established interface and interaction patterns found in social media and institutional digital measures software. Inside the interface, functionality that was similar in nature was grouped visually according to the principle of uniform connectedness [19]. Drafts of the design were reviewed by potential users at several points to confirm that the design was working as intended.

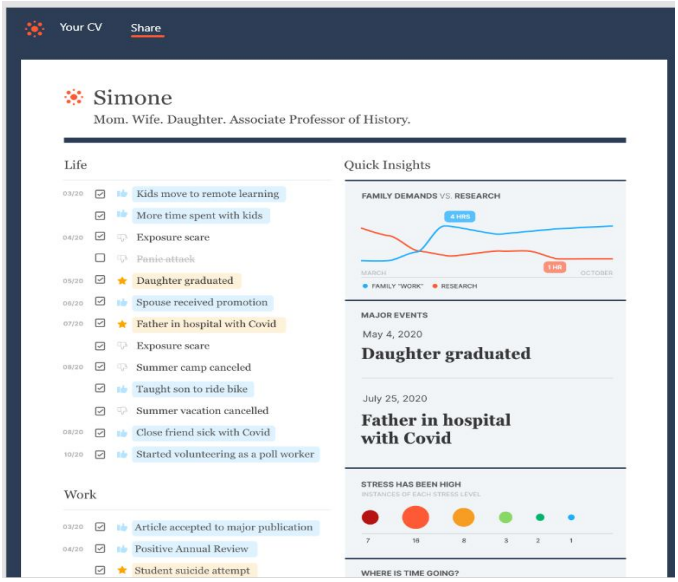

Figure 2: CovidCV Visual representation.

The conceptual user interface was designed using *Adobe XD* [20]. Instead of developing the HTML and CSS from scratch, a plugin called *Web Export* [21] was used while translating the XD design to the corresponding code. A sample representation of CovidCV, created using XD and implemented programmatically, is shown in Figure 2. Users can access their CV only after successful login and associated data input. An authorized user is allowed to download and share the CV with others - based on specific user preferences.

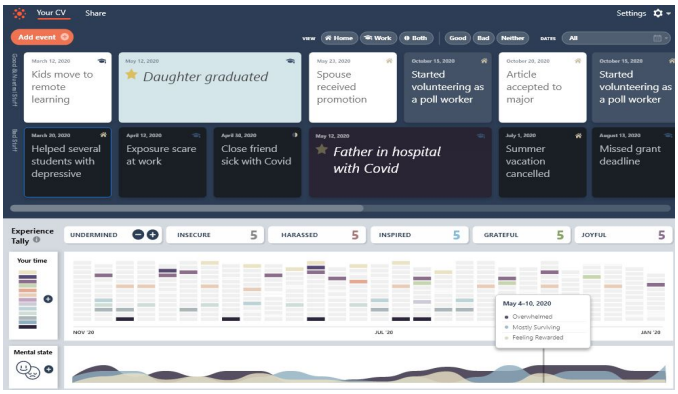

Figure 3: CovidCV - User Dashboard.

B. Login and Authentication Process

Authentication, a necessary feature for the CovidCV system, was designed using the *OAuth* standard 2.0 [22] [23] and is used to grant authorized access to third party applications on behalf of end users. In the CovidCV system, the *Resource owner password credentials grant type* of OAuth

is used – this is used when an application authenticates itself as well as the end-user. More specifically, using this grant type during the login process, the front-end sends its own credentials as well as user credentials to the back end. The back-end first checks if the credentials of the front-end (client) are valid. If they are, it checks the credentials of the end-user in the database, and if that check is successful, then the back-end issues an access token to the front-end.

A new user initially is requested to sign up. Once the signup process is successfully completed, the user can log in to the CovidCV system to provide their work and home life entries. A sample dashboard for a simulated user, containing representative entries, is indicated in Figure 3.

C. Type of APIs

In the CovidCV system, the REST standard [24] to is used to design the necessary APIs. Four different types of APIs are used: a) GET, b) POST, C) PUT, and d) DELETE. The GET method is used when the purpose of the endpoint is just to supply the necessary data without any state changes (e.g., displaying the entries on the dashboard). The POST endpoints are used when it is necessary to insert a new row to any table in the database (e.g., adding an user's activity). The PUT method is used when the purpose of the endpoint is to update a row in the database (e.g., updating an user's email). Lastly, the DELETE method is used when the purpose of the endpoint is to delete a row from the database for any specific reason. In such a case, the CovidCV system simply returns *success 200 status* and a *null* response.

D. Request handling

The application logic tier of the CovidCV system is further subdivided into four layers: a) Controller Layer (CL), b) Business Logic Layer (BLL), c) Service Layer (SL), and d) Repository Layer (RL). When a request arrives at the Tomcat web container, it is verified for all security and validity checks. If the request passes these checks, the Tomcat sends it to the CL for further processing. When the CL accepts the request from Tomcat, it simply forwards it to the BLL for further processing. The BLL performs additional validation checks, such as duplicate user's names during the signup process, and issues any necessary error messages. Once, all business logic checks are performed, the request parameters are forwarded to the SL. The SL is responsible for creating necessary objects using the parameters provided by the client. After creating the objects that correspond to database tables, they are forwarded to the RL. The RL performs queries on the database such as insert, update, delete and read. If connection with database is not available, the RL throws an error and advises the user to retry at a later time.

E. DTO vs. Payload vs. Model classes

In CovidCV system, there are three types of plain old Java objects (POJOs): a) Data Transfer Objects (DTOs), b) Payload Classes (PCs), and c) Model Classes (MCs). The purpose of the DTOs is to transfer data across multiple components in the CovidCV system or return the response of the request to the user. The DTOs are necessary because sometimes it is not necessary to return every column of the database entry to the user. For example, the password of a user is stored in the database, and that user wants to see his/her data. In this case, the back end should not transfer sensitive data to the client. Therefore, in this case, the DTOs are used to provide only the necessary fields to the clients. The purpose of the PCs is to get data from clients. For example, if a user wants to sign up into the CovidCV system, the back-end needs that user's specific data. The PCs are made part of the interface that is visible to clients in order to know what attributes are required for a request to get it processed successfully. Finally, the purpose of MCs is to emulate database tables in terms of Java objects to achieve object relational mapping. For any interaction with the database, these MCs are used. Therefore, for any table in database, a corresponding MC is defined in the back-end layer. When specific items are retrieved from the database using the JPA repositories, MCs are returned. Using these attributes, from the MCs, DTOs are created to supply only the necessary information to the clients (and eventually end users).

Table I: Timing breakdown of a single user request.

Timing Phases	**GET** (ms)	**POST** (ms)	**PUT** (ms)	**DELETE** (ms)
Resource Scheduling				
Queueing	42.32	58.5	3.52	3.05
Connection Start				
Stalled	23.37	0.94	1.85	1.3
DNS Lookup	-	52.6	50.26	-
Initial Connection	5.2	45.71	70.47	-
SSL	5.19	27.12	46.04	-
Request/Response				
Request Sent	0.39	0.53	0.36	0.51
Waiting (TTFB)	93.85	139.78	189.84	71.98
Content Download	24.15	2.29	2.01	1.32
Total	194.67	327.47	364.35	78.16

F. Scalability

In our experiments, scalability is measured by observing the relation between the number of requests and their average execution time for different types of requested APIs (GET, POST, PUT, and DELETE). We have experimented with the CovidCV system by implementing a test module for firing single user and multiple (three) user requests simultaneously. We compute the end-to-end response time for a user request by combining the individual times required by the front-end and back-end layers.

The POST, GET, PUT, and DELETE APIs correspond to the each of the CRUD operations, respectively. These timing

details, for a single user, are shown in Table I. The entries in Table I indicate that the *Create* and *Update* operations require more time than other operations. The total time required for each operation, as shown in Table I, is broken down into three individual times: Resource Scheduling time, Connection Start time, and Request/Response time. We have also conducted a similar experiment for multiple (three) simultaneous user requests. Apart from, changes in the Waiting (*TTFB - Time To First Byte*) and Queuing times, we did not see changes to the other individual times. The average waiting time and the average Queuing time, for all APIs combined, were 928.64 msec and 538.98 msec respectively. These two values contribute to the end-to-end response time – for multiple users, the average value of the end-to-end time was 1.48 sec; whereas for a single user, it was 241.2 msec.

V. Conclusion and Future Work

The CovidCV system is a web-based App for faculty and scholars to document their entire holistic life in the recent pandemic. It is intended to provide multiple functionalities: a) self-improvement, b) possible mentorship, and c) addressing inherent problem of inequality in academy. The CovidCV system is capable of storing and displaying many different life activities of the users unlike traditional CVs. The CovidCV system allows the user to record different events by categorizing them in various types (e.g., good, bad, or neither) and the corresponding locations (e.g., home, work or both). Any event can be marked as major or a regular event. The CovidCV system keeps track of the time spent by user on various activities. It also records for any week, the feelings and emotions of the user. All these details are provided via an effective and user friendly graphical interface. Currently, the CovidCV system is designed for faculty members and scholars as the intended users of the system. The CovidCV system has been internally deployed and preliminary feedback has been encouraging. Soon it will be used in a larger campus-wide study. Various extensions are possible to the CovidCV system. Specifically:

- To develop a admin portal to see a comprehensive and aggregated view of all users of the CovidCV system.
- To create a mobile-based front-end to the CovidCV system.
- To deploy multiple replicas of the back-end layer with a load balancer can be introduced to distribute load among replicas and provide necessary fault-tolerance.

VI. Acknowledgements

This project is supported in part by the IU Office of the Vice President for Research and the IUPUI Arts and Humanities Institute.

References

[1] S. M. Bianchi, L. C. Sayer, M. A. Milkie, and J. P. Robinson, "Housework: Who did, does or will do it, and how much does it matter?" *Social Forces, Vol. 91, Issue 1*, 2012.
[2] G. Viglione, "Are women publishing less during the pandemic? here's what the data say." *Nature 581, no. 7809*, 2020.
[3] J. L. Malisch, B. N. Harris, S. M. Sherrer, K. A. Lewis, S. L. Shepherd, P. C. McCarthy, J. L. Spott, E. P. Karam, N. Moustaid-Moussa, J. M. Calarco, L. Ramalingam, A. E. Talley, J. E. Cañas-Carrell, K. Ardon-Dryer, D. A. Weiser, X. E. Bernal, and J. Deitloff, "Opinion: In the wake of covid-19, academia needs new solutions to ensure gender equity," *Proceedings of the National Academy of Sciences*, 2020.
[4] C. Michelle, I, D. Natalie, and M.-W. Diana, "Preventing a secondary epidemic of lost early career scientists. effects of covid-19 pandemic on women with children," *Annals of the American Thoracic Society*, 2020.
[5] A. Vineet, M., W. Charles, M., O. Avital, Y., S. Mark, and S. Jain, "Using the curriculum vitae to promote gender equity during the COVID-19 pandemic," *Proceedings of the National Academy of Sciences*, 2020.
[6] J. L. Malisch, B. N. Harris, S. M. Sherrer, K. A. Lewis, S. L. Shepherd, P. C. McCarthy, J. L. Spott, E. P. Karam, N. Moustaid-Moussa, J. M. Calarco, L. Ramalingam, A. E. Talley, J. E. Cañas-Carrell, K. Ardon-Dryer, D. A. Weiser, X. E. Bernal, and J. Deitloff, "Reply to Arora et al.: Concerns and considerations about using the CV as an equity tool," *Proceedings of the National Academy of Science*, 2020.
[7] "The American Council of Learned Societies Awards 45 Emerging Voices Fellowships," https://www.acls.org/ACLS-News/ACLS-News/August-2020/The-American-Council-of-Learned-Societies-Awards-45-Emerging-Voices-Fellowships/?category=fellowshipandgrantcompetitions.
[8] "A JavaScript library for building user interfaces," https://reactjs.org.
[9] "Node.js®, A JavaScript runtime built on Chrome's V8 JavaScript engine," https://nodejs.org/en/.
[10] "ngnix," https://www.nginx.com.
[11] "Spring Boot, Create stand-alone applications," https://spring.io/projects/spring-boot.
[12] "Spring Security, a powerful and highly customizable authentication and access-control framework," https://spring.io/projects/spring-boot.
[13] "Hibernate Architecture," https://www.javatpoint.com/hibernate-architecture.
[14] "Apache Tomcat. Open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies," http://tomcat.apache.org/.
[15] "Apache Maven," https://maven.apache.org/.
[16] "MySQL," https://www.mysql.com/.
[17] "Database column-level encryption with Spring Data JPA," https://sultanov.dev/blog/database-column-level-encryption-with-spring-data-jpa/.
[18] J. Nielsen, "Jakob's Law of Internet User Experience," *https://www.nngroup.com/videos/jakobs-law-internet-ux/*, 2017.
[19] H. S., H. G., W., and C. L., "Uniform connectedness and classical gestalt principles of perceptual grouping," *Perception Psychophysics*, 1999.
[20] "Adobe XD," https://www.adobe.com/products/xd.html.
[21] "Web Export ," https://velara-3.gitbook.io/web-export/.
[22] "OAuth 2.0," https://oauth.net/2/.
[23] "The OAuth 2.0 Authorization Framework," https://tools.ietf.org/html/rfc6749.
[24] "HTTP Methods," https://restfulapi.net/http-methods/.