Designing Crowdsourcing Software to Inform Municipalities About Infrastructure Condition

Eric Shoemaker, Harrison Randolph, James Bryce and Husnu S. Narman
{shoemaker30,randolph69,bryce,narman}@marshall.edu
College of Engineering and Computer Sciences, Marshall University, Huntington, WV 25755

***Abstract*—Determining how to utilize and allocate budgets for public infrastructure is an essential issue that local governments face. For a beneficial budgeting plan to be made for civil engineering infrastructure maintenance and projects, a local government needs first to know the condition of its infrastructure. Without such information, it is not possible to create a plan that will use funds effectively and address the needs and desires of the community members. For larger bodies, like state governments or populous cities, collecting this information is achievable through the dedication of workforce and surveyors. However, smaller municipalities can often lack the funds and the resources. Consequently, this will foster reactive management of public infrastructure such that information is collected after an issue occurs and becomes major rather than minor, so repairs can be more expensive if the damage is significant. This only applies more strain to the limited budgets. To solve this issue, crowdsourcing information about the condition of public infrastructure from the community in the municipality can be used. Therefore, we developed a crowdsourcing based system to address this issue. The data will be collected through the use of public software that will seek to collect data and encourage users to report about all issues—, not just severe or significant damages. By receiving reports about all issues, regardless of severity, municipalities will have the information not only to react to and fix the damage, but also they may proactively predict, repair, and prevent severe damages based on timelines of reports about more minor issues. Moreover, upon the adaptation of this method proves to be helpful, this research aims to produce and deploy such software to benefit smaller municipalities.**

***Index Terms*—crowdsourcing, design, reporting, infrastructure**

I. Introduction

Civil infrastructure is an imperative part of all societies, and it requires a significant amount of social resources [1]. It is the glue that holds together the cultural, environmental, and social structures that create a more civilized community. Consequently, smart infrastructure investment and designs enhance connectivity, productivity, and satisfaction within a community. Therefore, when a city is engaging in smart infrastructure design and investments, it is more likely to see economic success and to attract new residents to keep its economy growing [2]. This is very important in an age where cities are competing for investments, talent, entrepreneurs, and young families [3].

However, making beneficial decisions regarding the use of allocated funds for public infrastructure is one of the most important problems that local governments face [4]. It requires high-cost decision-making and demanding amounts of time and research in order to determine the locations and severity of public infrastructure damage [5]. Along with this, community input should be gathered and considered before the development of new businesses and infrastructure into a community. For a beneficial budgeting plan to be made, the local government needs to know the condition of its infrastructure and the opinions of members of the community. Although larger bodies such as state governments and populous cities have the resources to dedicate a workforce to the collection of data regarding the condition of their infrastructure, geographically smaller and less populous municipalities may not have the funds and/or resources [6].

For small municipalities, a method of solving this issue is to utilize crowdsourcing to gather this data from people living in the municipality. If large amounts of community involvement occur, crowdsourcing would allow for enough data to be collected at low-cost [7], and it essentially gives access to a large group of potential workers, all of whom have a diverse range of skills and experiences within the municipality [8]. In order to utilize crowdsourcing for this purpose, a means for those living in the municipality to inform their local government of issues with public infrastructure must be established and commonly used.

Generally, issues with public infrastructure are reported by individuals to the local government owning the infrastructure, and there are many available methods for anyone to create such reports. Many cities utilize websites, emails, and phone lines that allow the community to inform them about public safety hazards and damages. Also, government infrastructure organizations in each state, such as the Division of Transportation [9], provide the public with forms with which to report roadway issues—such as potholes—on their website, and they allow people to create claims for reimbursement if these issues are the cause of damage to personal property.

Besides these options, there have been efforts to enhance convenience and connectivity to those living in cities of all sizes. Applications—[10], [11], [12] for examples—have been developed to allow users to easily and conveniently report damaged public infrastructure using a phone app. These applications allow for individuals to submit these issues to their local governments and allow those living in communities to see other's reports and to be informed of the hazards and damages around them. They also provide the community with tools to up-vote others' reports to make an effort to enhance the relationships of the local government and its people based on consideration of and response time to the issues presented

by the people.

All of these options successfully allow for the public to inform the government about public infrastructure, which is a goal of this paper, but the *uniqueness* of this paper is to supply data such that local governments can make beneficial budgeting decisions for their public infrastructure. Therefore, simply collecting reports from users after damages have occurred may not be sufficient. Users should be encouraged to report any damage regardless of its impact or severity, so local governments have sufficient data to be proactive and address damages in earlier stages. Along with this, users need to be encouraged to report often and whenever they see potential issues rather than only reporting about damages that have affected them already.

Therefore, the *objectives* of this paper are to analyze the usefulness and effectiveness of crowdsourcing as a low-cost option of data collection for small municipalities and to design software such that local governments will receive information in a format that allows them to proactively—, not just reactively—manage their public infrastructure.

The key *contributions* of this paper are:

- Software, which allows users to conveniently report about public infrastructure and to inform local government, is designed and tested.
- The ability of the software to collect meaningful and detailed reports while maintaining convenience and speed is analyzed through the results of small-scale application deployment and analyzing each submitted report.
- The overall effectiveness of the collected data in documenting the infrastructure condition of an area and the ability of such data to be used in proactive infrastructure management is analyzed during the deployment.

The *results* of the small-scale deployment of the software design showed that, on average, it takes new users 140.38 seconds to complete a report and returning users only an average of 78.15 seconds. Along with providing a short reporting time, it is determined that 82% of reports provide an accurate location, 97% of reports effectively convey the type of damage and its severity, and 100% of reports are able to document the time of the report such that a timeline of the damage could be created. Overall, 93% of reports can be easily used for proactive management.

Along with these things, user feedback shows that 89% of users who leave feedback favor the use of a questionnaire rather than a required written description to accelerate the reporting time. Also, 5% of users inform of confusion or issues with the design of the reporting software.

The remainder of this paper is organized as follows: Section II discusses the reasoning for the initial software design. Section III discusses the system design and the structure of a damage report. Section IV discusses a small-scale deployment of the crowdsourcing software and analyzes the successes and failures of the software design. Section V concludes the research and discusses future work and plans for the software.

II. Determining Effective Design for Crowdsourcing Software

To provide a method of crowdsourcing data for a small municipality, it is determined that software can be developed and makes easily accessible to every individual living within the municipality. Before development for the software, existing software to collect such information, such as reporting forms for potholes on public streets, have been examined for both effectiveness and shortcomings. Consider the damage to a public roadway, such as a pothole, at Marshall University in Huntington WV, USA. When someone observes the damage and intends to report it, the individual will utilize one of the available reporting options. One option is to visit the website for the City of Huntington in order to report the pothole, but one will only find a web page [13] instructing him/her to call the Public Works Department. Alternatively, one can submit a written description of the damage to one of the emails provided on the web page. This method can quickly fail to inform the local government about the issue if the individual reporting leaves insufficient information about the damage or its location. The other method the individual could consider would be to use the WVDOT website [14] to report the damage to the state government. While this method provides a reporting form that will effectively collect the location of the damage from the user, it still relies on a thorough and accurate description from the person writing the report. Along with this, the form requires much unnecessary personal information, such as name, email, and mailing address.

Both of these methods share similar issues, and they contribute directly to reactive management of infrastructure rather than proactive. It is likely that this occurs for two reasons. First, many people do not worry about an issue until it affects them directly. It is unlikely for someone to be concerned about a developing issue unless it causes an issue for them. For example, many people will not take the time and effort required to report a pothole unless the pothole causes a significant inconvenience, such as causing damage to one's vehicle, and the individual's notification to the local government will more likely be in the form claim of damage to personal property rather than an informational report regarding the pothole itself. Second, many of the existing methods of reporting public infrastructure damages are inconvenient and/or confusing. As stated before, the WVDOT reporting form [14] requires much personal information, and it asks repetitive questions. For example, users are instructed to enter the county, route number, road name, nearby landmarks, etc., but the user also is instructed to click the exact coordinates of the damage on a map. For these reasons, an individual on-the-go may decide not to report observed issues to the local government even if the issue is obvious and significant.

These issues matter and they continue to promote reactive management of infrastructure by causing many users only to report major issues after the fact, so in order for crowdsourcing software to promote proactive management, these issues will need to be addressed in the design process. In order to

encourage users to use the software more frequently and for less significant issues, the design of the software must value convenience highly. It should be easily accessible to anyone at any time, and completing a report should take very little time. The user interface must be inviting and simple to effectively encourage all users, regardless of age or technological experience, to enjoy creating reports, and this assures that users are more likely to return to the software again [15]. The software must also assure users that their reports are being efficiently collected, presented, and considered by the local government. If these elements are all executed well, users may be more likely to report less significant issues. Catching issues at early stages in damage significantly help municipalities to prevent worsening of the issue into one which is severe and/or hazardous, and in doing so, potential injuries and further damages can be prevented [16].

However, this poses another issue. If the reporting process is made with too much focus on fast reporting times and convenience, less-detailed reports may be submitted. The more detailed each report is, the more effective its information can be used to help a municipality in making budgeting decisions, and without reports that are easy to interpret, categorize, and organize, the software will lose its ability to be meaningful and effective. Therefore, the existing process for such software where users create written descriptions may prove to be ineffective.

Therefore, the design of this software must find the balance between reporting speed and reporting detail. To find this balance, it is determined that a questionnaire can be provided to users instead of requesting written reports. This way, a user can file a report with only a few clicks. Along with this, automation can be used to accelerate this process. For example, if the user is to report an issue when she/he sees it, the location and time can be recorded automatically.

Using the above principles, the software is designed to accomplish this goal.

III. System Models

In this section, the system's design, organization methods for the database of information, and the structure of a report are introduced.

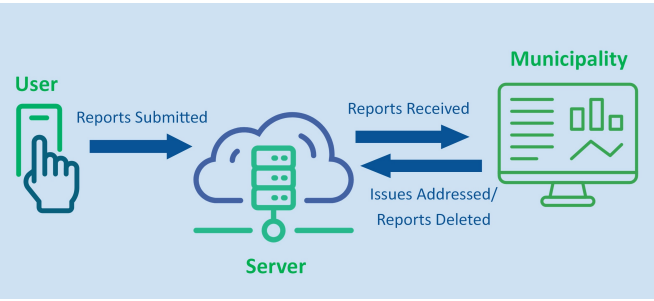

Fig. 1. System design to collect data through crowdsourcing.

Fig. 1 shows a model of the structure of the software system. The server utilizes a full LAMP-stack design to provide crowdsourcing software to the public, store all answers to public reports, and handle requests for data from municipalities.

The software available to the public is a website providing a form with which to create a report. When a report is completed, the data is sent to the server. The server holds all of the reports in a database system accessible by a local government, and that local government may use this information when developing budgeting plans for public infrastructure, making repairs in the early stages of damage, and addressing major damages.

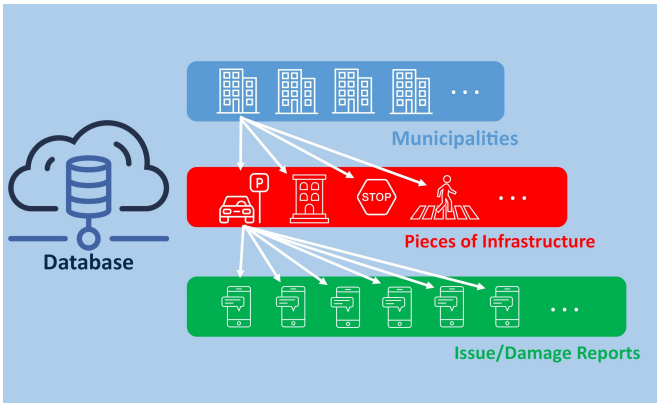

Fig. 2. Organization of the Database System.

Fig. 2 shows the organization of reports in the database system. Each local government to access the system may navigate through a database of every piece of infrastructure that has ever received a report within the governing body. Upon accessing a piece of infrastructure, a timeline of every report—and each time a report is addressed/repairs are made—may be accessed. This structure aims to encourage proactive management of infrastructure by making patterns of issues completely obvious to observers.

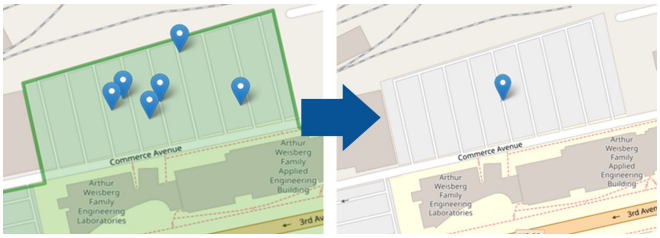

Fig. 3. Update to organization of database storage and access.

This database design is adopted for this system after analyzing results from the software's deployment (discussed in Section IV). The previous design of this software —as shown in the left of Fig. 3—shared similarities with existing applications such that it would provide local governments with maps and lists of every report. This design failed to encourage proactive management because the progression of damage documented from many reports may not be obvious to observers. Therefore, instead of showing maps of every

report, in the current design, the maps now show each piece of infrastructure, and upon selecting one, a chronological timeline of reports will be accessible in list format. This way, the observer will see the lifespan of a piece of infrastructure leading up to the most recent damage.

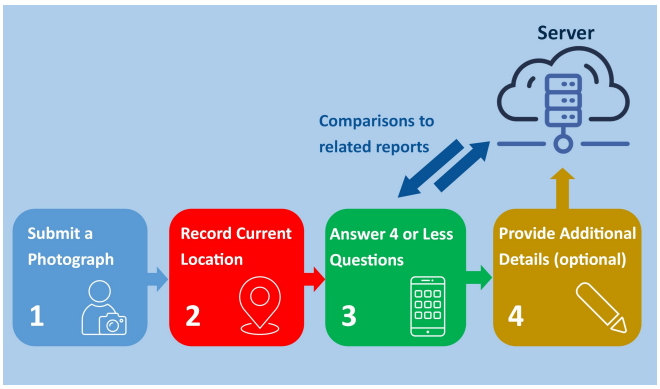

Fig. 4. Structure of an Infrastructure Damage Report

Fig. 4 shows the structure of a damage report. When a user begins creating a new report, the user is instructed to complete it at the location of the issue being reported and then prompted to take a photograph of the issue. Through additional options, the user can see whether a picture is necessary for the issue being reported. Usually, a photo is required unless obtaining the photo could put someone at risk, such as obtaining a picture of a pothole on a busy street.

After obtaining the photo, the device's current location is used for reporting —with the user's consent —is recorded automatically. The user is then presented with the first multiple-choice question. These questions are dynamic depending on user answers, and there can be a maximum of four questions. These questions are optimized such that the entire report should take about a minute to complete, and the reporting application communicates with the server to determine which questions best narrow down the issue quickly. The flow of the questions is organized using a decision tree [17].

The first question asks about the type of infrastructure that is being reported; for example, users will answer roadway, sidewalk, bridge, building, etc. Upon answering this question, the server compares the answer and the location to other reports, and if it finds matches, it presents the user with a photo from a recent report and asks the user if he/she is reporting about this piece of infrastructure. If so, the report is associated with an existing piece of infrastructure in the database and will be added to the timeline. If not, then this piece of infrastructure will be added to the database, and this report will begin the timeline of reports about the piece of infrastructure.

The remaining questions serve to narrow down the specific issue and its severity quickly. For example, if the user selects a sidewalk, the second question will give options related to sidewalks to choose from, such as cracking, water accumulation, etc. Then, the user may be prompted to rate the severity of the issue or its posed safety risk to the public on a scale of one to five.

At this point, the user may submit the report, and the report should contain sufficient details such that it can be easily interpreted and used. However, in some specific cases, more information may be necessary. For example, if the user reports about damage inside a public building, additional written details may be necessary for the location of the problem to be located inside. The GPS information will likely be insufficient, so users will be asked to leave brief information about the room.

IV. Deployment and Analysis of Results

In January 2021, the crowdsourcing software was made available to students at Marshall University. The university would simulate a small municipality, and the researchers would serve as the local government of the municipality. Students were able to access the website throughout the Spring 2021 semester, and they were able to report about any infrastructure on Marshall University campus. This deployment aimed to analyze the effectiveness of the software design, to determine if the software effectively collects and organizes data in a meaningful and useful way, and to determine if reports are both completed quickly but detailed enough to be used efficiently and proactively

Throughout the semester, the speed at which students completed reports were analyzed. To accomplish this, a timestamp was recorded along with each user's answer submitted. This allows us for a complete analysis on which sections of a report caused users the most issues, and it also informs which parts of the report went well for users. Along with these things, it assists us to whether the overall reporting process is efficient or if it needs further improvements in performance or structure.

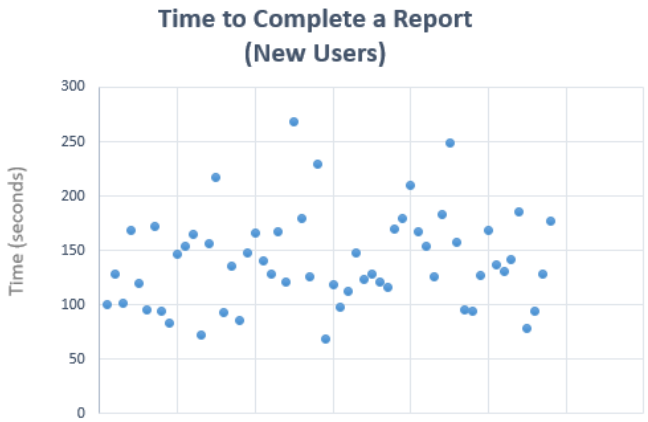

Fig. 5. Scatter plot to represent the amount of time it takes for new users to complete a report.

Fig. 5 shows the total amount of time it takes a user to complete his/her first report. If a user does not complete a report at all or is idle for more than 90 seconds while completing a report, the result is not included since these results do not accurately reflect the actual amount of time

it takes to complete a report. On average, it takes new users 140.38 seconds to complete a report. The design of this report aims to make reports take about 90 seconds since a faster reporting time will likely attract users to return to the software again.

This concern related returning to software to report other issues are observed to be valid because only 23% of previous users returned to the software to complete a second report. To find the root cause of this issue, the timestamps of users who had already completed a report before were analyzed, and the percentage of users who returned was measured.

Fig. 6. Scatter plot to represent the amount of time it takes for returning users to complete a report.

Fig. 6 shows the total time that it takes for returning users to complete a report. As expected, returning users are able to complete reports more quickly. The average time to complete a report for these users is 78.15 seconds which is under the aim of 90 seconds. This is a shorter amount of time than anticipated and shows that the report design has the potential to be completed quickly.

Because of this, it is determined that the primary reason that new users take much longer to complete reports is explicitly due to the interface design and provided instructions. The questions and answer choices provided to users are not sufficiently self-explanatory, and it takes new users longer to understand what is asked of them. Along with this, users may have experienced issues navigating the web form due to browser incompatibilities, having JavaScript disabled, etc. These are issues that will need to be addressed in the future to ensure users take enough interest in this project to return to the software.

As discussed previously, this study aims to find a balance between having a short reporting time and ensuring that reports are detailed enough to be used effectively, and it proves to be successful. Along with providing users with a short reporting time, most reports include a sufficient amount of details for municipalities to use.

Fig. 7 shows the percentages of valuable data from the submitted reports. Here, valuable data refers to data that completely and obviously portrays an answer. Regarding finding the damage location, 82% of the reports obtained clearly expressed the location within 30 feet. The remaining 18% consisted of users who either reported from the inside of a building or those who misunderstood the instructions and reported about an issue they were not near. Several of those reporting from inside a building clearly described the problem when prompted but failed to provide any necessary details regarding the location. For example, a report was received requesting more visibility of yellow paint on a stairway. However, the specific stairway was never mentioned, and the user's GPS gave a location outside of the building which was not near any staircase. Reports regarding damage to roadways were not considered in this percentage because getting within 30 feet of the damage could be dangerous to users, so users were specifically instructed to stay safe and avoid walking on roadways. Because of this, some users did not submit a photo or an accurate location.

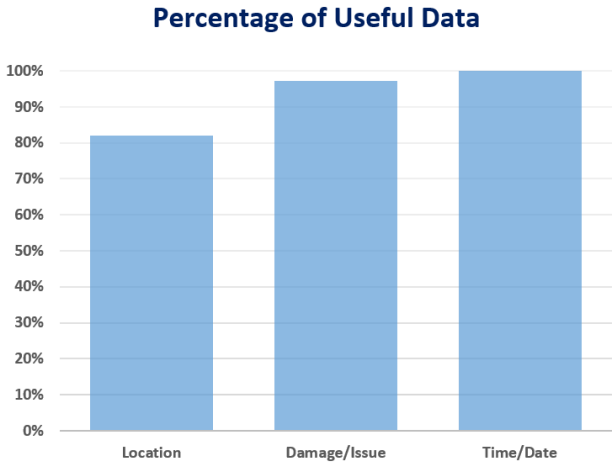

Fig. 7. The percentage of which data collected are useful and informative for local governments to interpret.

Regarding the damage itself, 97% of reports effectively communicated the issue in an unambiguous way. Answers to the questionnaire definitively defined the exact issue, and severity ratings from users and the submitted photo showed the exact severity of the damage. The remaining 3% consisted of reports where the issue fell outside of the available options for questions. For example, if a user reported damage to a public bench, they would choose either 'Sidewalk' or 'Other' when asked what piece of infrastructure was being reported. Although descriptions and photos easily and quickly narrow down the issue, they may be ambiguous depending on the description provided by the user.

Regarding the time and date, 100% of the reports gave appropriate details such that the reports could be organized chronologically. Because this process was automated, it was expected that a high amount of reports would accomplish this. However, user misunderstanding and long idle times —from users leaving the web page open and finishing a

report later date or time —could have addicted this percentage. Safeguards, such as time-out features, will be implemented in the future. From this data and further inspection of the submitted reports, it is determined that a total of 93% of the reports are overall useful and complete.

Along with analyzing reports for their speed and detail, user feedback about the software is addressed. Regarding the design, users provided much feedback about slight changes to enhance convenience and design, and this feedback will be addressed. Along with this, there was much feedback about the system design itself. Because the system utilized a website, users with poor internet connections faced inconveniences. Also, several browser compatibility issues were reported. For example, those using the Microsoft Edge browser were unable to save reports to the database. When the user's coordinates are recorded, an API call is made in order to reverse-geocode the location. Then, this information is used to appropriately place the report in the database for the correct municipality. The browser can be incompatible with the API call made, thus throwing confusing errors to users about MySQL insertion.

The design received positive feedback as well. Of the users who commented about the design of a report, 89% of users liked the design of a questionnaire system to accelerate the reporting process when compared to written reports as used by existing sites [14]. Other users commented about the appearance of the site and accessibility specifics. Most accessibility features, such as text-to-speech, are handled by the device and browser used for reporting, but issues like small text size should be updated.

V. Conclusion and Future Work

In order for a municipality to create a beneficial budgeting plan for infrastructure maintenance and projects, it must first know the condition of its infrastructure. However, small municipalities may lack the funds and/or resources to gather such information, so to provide such municipalities with this information, the data is gathered from those living in the municipality through software utilizing crowdsourcing. Along with gathering data, the software intends to encourage proactive management of infrastructure by encouraging users to report about all issues—regardless of severity at that time—and presents the information in a meaningful way. This way, municipalities may recognize patterns and conclude timelines of damage progression. This software was developed and deployed in a case study, and it is designed to be both fast and thorough through the use of a decision tree rather than through written descriptions.

Our proposed software and crowdsourcing technique are proved to be successful but need several improvements to be considered in future work. It is apparent from reporting times and feedback that instructions need to be made clearer, and browser incompatibilities and network issues will need to be addressed. This issue will be solved through the redevelopment of the reporting software into a native mobile application. This way, different browsers cannot change a user's experience through varying CSS or script interpretation [18], and the reporting software will not have to be fetched from the server. Communications with the server can occur through asynchronous API calls, allowing for a seamless user experience even at slower network speeds. Besides these things, further studies about public acceptance and popularity of the application will be conducted.

Acknowledgement

This work was funded in part by the NASA West Virginia Space Grant Consortium and in part by the Marshall University Research Corporation under the Undergraduate Creative Discovery and Research Scholar Award Program.

References

[1] W. Buhr, "What is infrastructure?" Volkswirtschaftliche Diskussionsbeiträge, Tech. Rep., 2003.

[2] E. Ivanova and J. Masarova, "Importance of road infrastructure in the economic development and competitiveness," *Economics and management*, vol. 18, no. 2, pp. 263–274, 2013.

[3] H. M. Treasury *et al.*, "National infrastructure plan 2011," *Infrastructure UK, UK Treasury Department (HM Treasury), http://cdn. hm-treasury. gov. uk/national_infrastructure_plan291111. pdf*, 2011, accessed: 2021-04-25.

[4] W.-C. Huang, J.-Y. Teng, and M.-C. Lin, "The budget allocation model of public infrastructure projects," *Journal of Marine Science and Technology*, vol. 18, no. 5, pp. 697–708, 2010.

[5] A. Maji and M. K. Jha, "Modeling highway infrastructure maintenance schedules with budget constraints," *Transportation research record*, vol. 1991, no. 1, pp. 19–26, 2007.

[6] U. Wiberg and I. Limani, "Intermunicipal collaboration: a smart alternative for small municipalities?" *Scandinavian Journal of Public Administration*, vol. 19, no. 1, pp. 63–82, 2015.

[7] D. C. Brabham, *Crowdsourcing*. Mit Press, 2013.

[8] M. Baum, Neil, "Crowd sourcing," *The Journal of Medical Practice Management: MPM*, vol. 31, no. 4, pp. 238–239, Jan 2016, copyright Greenbranch Publishing, LLC Jan/Feb 2016; Last updated - 2016-07-13. [Online]. Available: https://www-proquest-com.marshall.idm.oclc.org/scholarly-journals/crowd-sourcing/docview/1803510742/se-2?accountid=12281

[9] "U.S. Division of Transportation," https://www.transportation.gov/, accessed: 2021-05-18.

[10] "Commonwealth Connect," http://commonwealthconnect.io/#home, accessed: 2021-06-05.

[11] S. Burgart, "Gap trap: A pothole detection and reporting system utilizing mobile devices," 2014.

[12] M. Foth, R. Schroeter, and I. Anastasiu, "Fixing the city one photo at a time: Mobile logging of maintenance requests," in *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, ser. OzCHI '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 126–129. [Online]. Available: https://doi.org/10.1145/2071536.2071555

[13] "Potholes," https://www.cityofhuntington.com/i-want-to/report/potholes/, accessed: 2021-05-28.

[14] "WVDOT Request for Road Repair Form," https://dotforms.wv.gov/cra/, accessed: 2021-05-19.

[15] J. Johnson and J. Jeff, *GUI bloopers: don'ts and do's for software developers and Web designers*. Morgan Kaufmann, 2000.

[16] Y. Fu, T. Hoang, K. Mechitov, J. R. Kim, D. Zhang, and B. F. Spencer, "Sudden event monitoring of civil infrastructure using demand-based wireless smart sensors," *Sensors*, vol. 18, no. 12, p. 4480, 2018.

[17] "Decision Tree V2," https://cimapp.org/downloads/DecisionTreeV2.pdf.

[18] J. G. Ochin, "Cross browser incompatibility: reasons and solutions," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 2, no. 3, pp. 66–77, 2011.