Quick Report - A web application that allows users to report events

Daniel-Octavian Popescu, Dragos-Gabriel Unguru, Petre-Bogdan Serban, and Stefan-Florin Paicu

ABSTRACT

Bureaucracy is starting to be a bigger problem in this era and people have a need to make these processes quicker. Quick Report is a web application that allows citizens to report new events from their community developed in ReachJs, NodeJs and Express with MongoDb for database and SeleniumAPI for testing. Users have to authenticate in the platform and then, they can report new events for state authorities. After the event is concluded, state authorities' representatives can change the status of the event and users can see what happened to that event.

Introduction

In Romania is a constant need for upgrading and creating web platforms to ease the communication between the citizens and the state authorities. The work that has been done in the last years showed that this will help with the bureaucracy and can lead to smaller queues and to a better communication. An automated platform where people can upload their new discovered events can lead to quicker fixes of those and it is a great tool for state authorities to find out their priorities.

Tech Stack

For creating this project the team worked with MERN tech stack in the developing phase:

- ReactJs - front-end
- NodeJs and Express - back-end
- MongoDb - database
- SeleniumAPI - testing

These technologies offer great tools to develop web application based on http request. For this project, the application will not be deployed on cloud, it will run on a local machine for the demo. Deployment will be discussed in future work section.

ReactJS is a JavaScript toolkit for creating reusable user interface components that is declarative, fast, and versatile. It's an open-source, component-based front-end library that's just responsible for the application's view layer[1]. It was created and maintained by Facebook and later utilized in Facebook products such as WhatsApp and Instagram. ReactJS allows the developers to completely separate the front-end from the back-end and use different languages for these two.

Node.js is a JavaScript run-time environment for servers. Google's V8 engine, libuv for cross-platform interoperability, and a core library are all open-source. Because Node.js does not operate in a browser, it does not offer a global "window" object. Without establishing a new thread for each request, a Node.js software operates in a single process[2]. Node.js' standard library includes a set of asynchronous I/O primitives that prevent JavaScript code from blocking, and libraries in Node.js are often created following non-blocking paradigms, making blocking behavior the exception rather than the rule. Instead of halting the thread and wasting CPU cycles waiting for a response, when Node.js conducts an I/O action, such as reading from the network, accessing a database, or accessing the filesystem, Node.js will restart the operations as the response comes back. This allows Node.js to handle thousands of concurrent connections with a single server without adding the overhead of thread concurrency management, which may be a major cause of errors.[3]

Express is a Node.js web application framework that offers a comprehensive range of functionality for both online and mobile apps[4]. Express offers tools for a robust routing for the endpoints of the application, offers tools that helps with working under the http protocol like redirection, and it is focused on creating high-performance web applications.

MongoDB is a NoSQL database management system that is free and open source. NoSQL is a database technology that is used as an alternative to traditional relational databases. When working with massive amounts of scattered data, NoSQL databases come in handy. MongoDB is a database that allows you to organize, store, and retrieve document-oriented data. MongoDB works using records, which are documents that contain a data structure made up of field and value pairs. MongoDB's basic data unit is the document. The documents resemble JavaScript Object Notation, although they employ a binary JSON variation (BSON). BSON has the advantage of supporting a wider range of data types. These documents have fields that are equivalent to columns in a relational database. According to the MongoDB user handbook, values can be a number of data formats, including other documents, arrays, and arrays of documents. A main key will be used as a unique identifier in documents.[5]

Selenium is an umbrella project encapsulating a variety of tools and libraries enabling web browser automation.[6]

Design and Preparation

This project started with creating the design for the pages of the application and creating a plan for development. The application targets citizens of a country and employees of different state authorities that can change the status of an event. For this, two different types of users are used throughout the application. The design of the application consists of 3 pages(first page, dashboard page, and insert event page). First page needs to have two buttons, one for login and one for register, and each one will redirect the user to the specific action 1. Dashboard page needs to show the user events inserted by him, if the user is a citizen one, and the events corresponding to the state authority, if the user is a state authority. Also, for the citizen user, a button is needed to redirect to the form that is used to create a new event 2. Insert event page consists of a minimalist form in which a title, a description and a category of the event is required 3. Proto.io was used for the design of the pages.

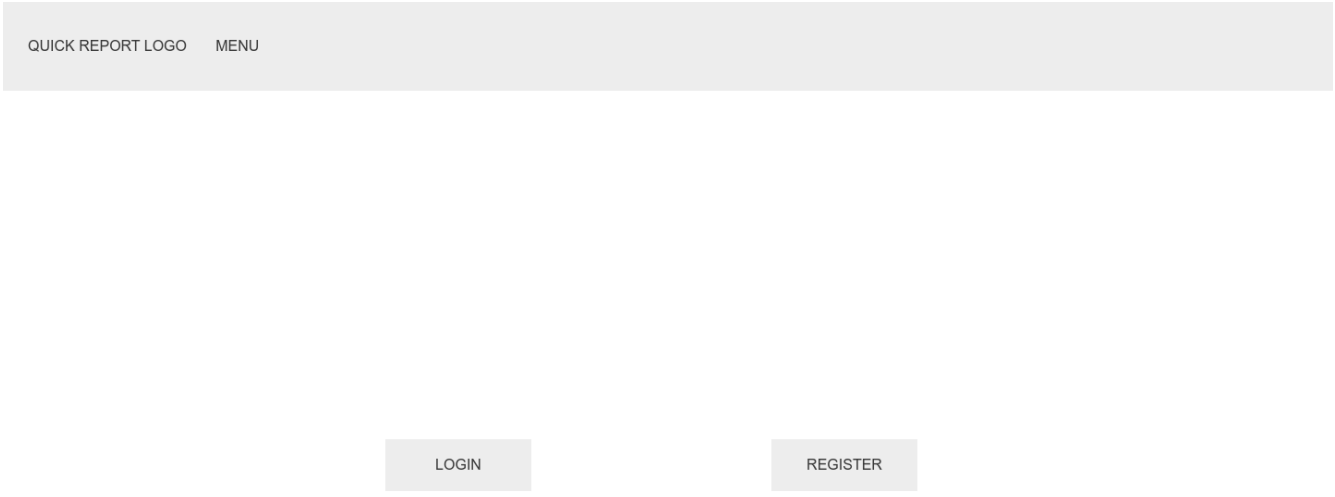

Figure 1. Start page

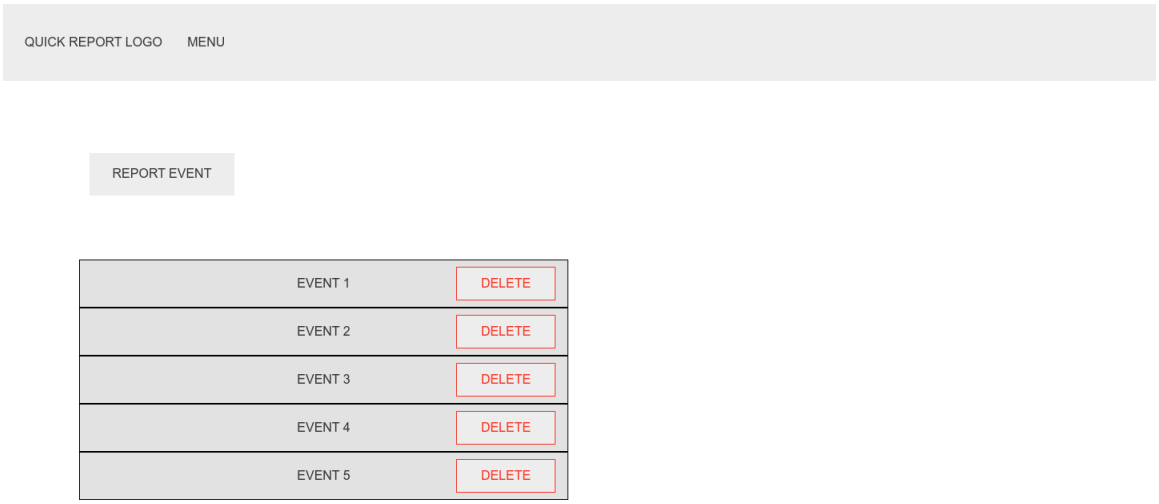

Figure 2. Dashboard

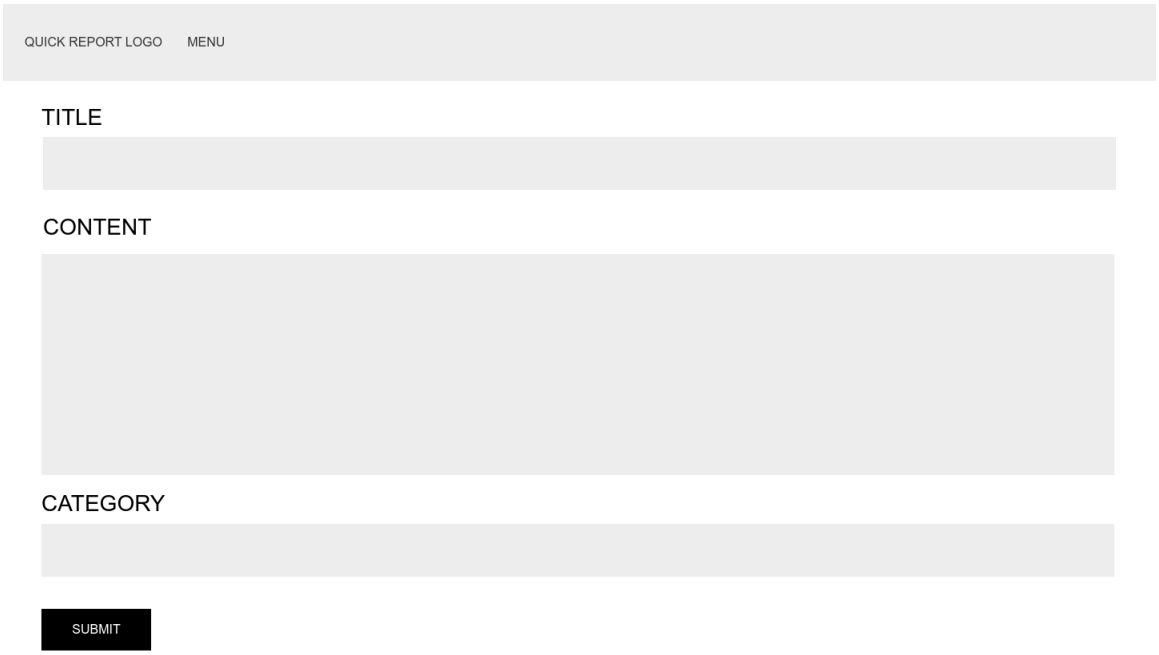

Figure 3. Insert Event

The authentication will be created using email and password, and it will use jwt tokens to post or retrieve data from backend. Data will be stored in MongoDB database, in a table specific for users. Another table for events will be created, each event having a user assigned. The backend in NodeJS and Express will provide the endpoints to post and get users and events and the frontend in ReactJS will help create the components that will display the information.

User Interface

The process of designing user interfaces in software or digital devices with a focus on appearance or style is known as user interface (UI) design. Designers strive to design user interfaces that are simple to use and enjoyable. UI design encompasses both graphical and non-graphical user interfaces, such as voice-controlled interfaces.

In this project, the user interface is minimal and user friendly, easy to work with, and shows explicit messages to users.

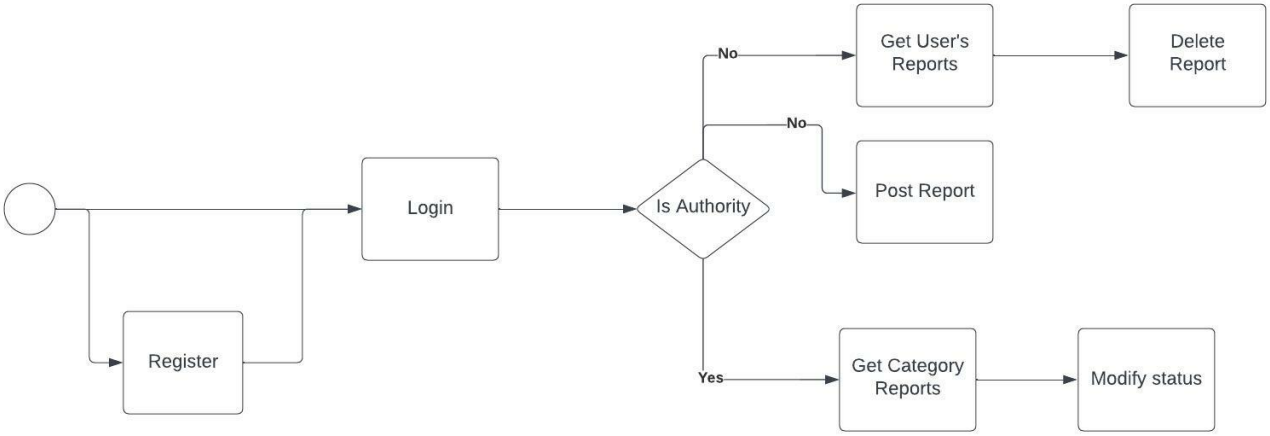

Figure 4. Flow Diagram.

When a user enters the application, will have the options to login if he already has an account, or to register if the user is new to the platform. As citizen users, they can add a new report to the platform and can see all of their previous reports. They can check every report status, having values of: created, in progress, solved. User have the option to delete a report. As authority users, they can see all the reports that correspond to their authority, can see the details of the reports, and they can change the status of a report, change that will be displayed to the citizen user's page.

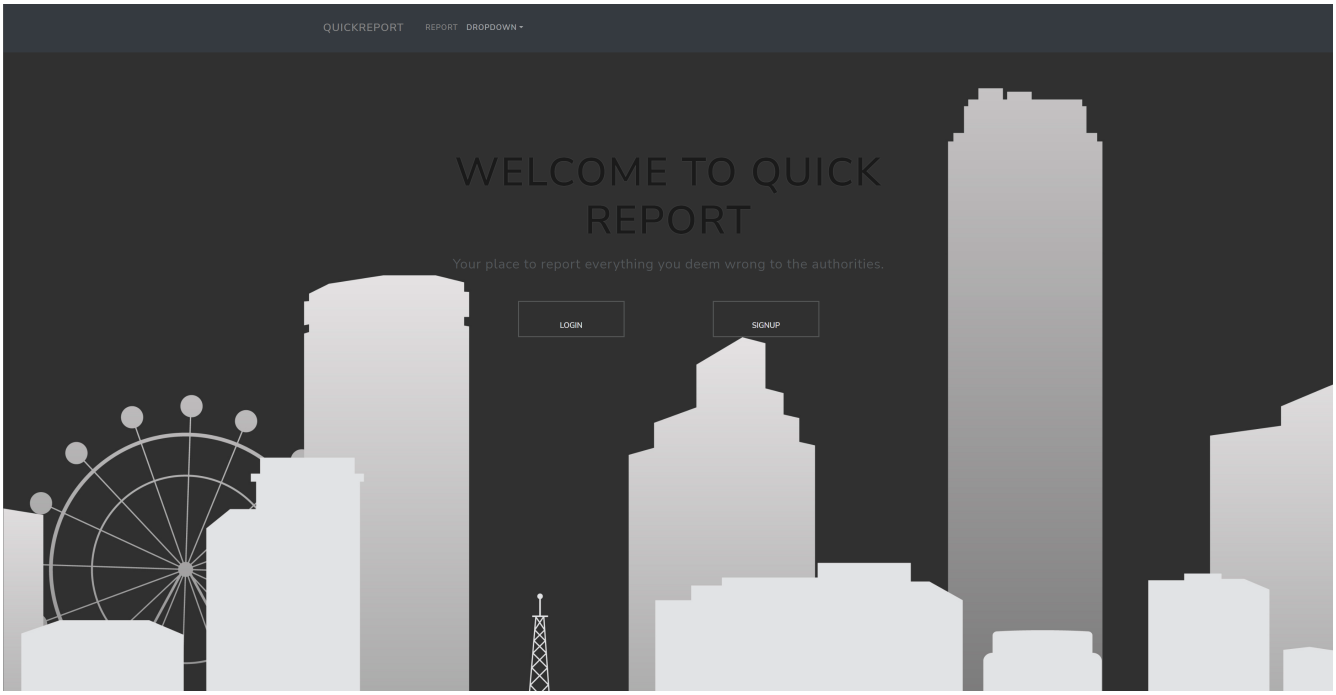

Figure 5. Start page

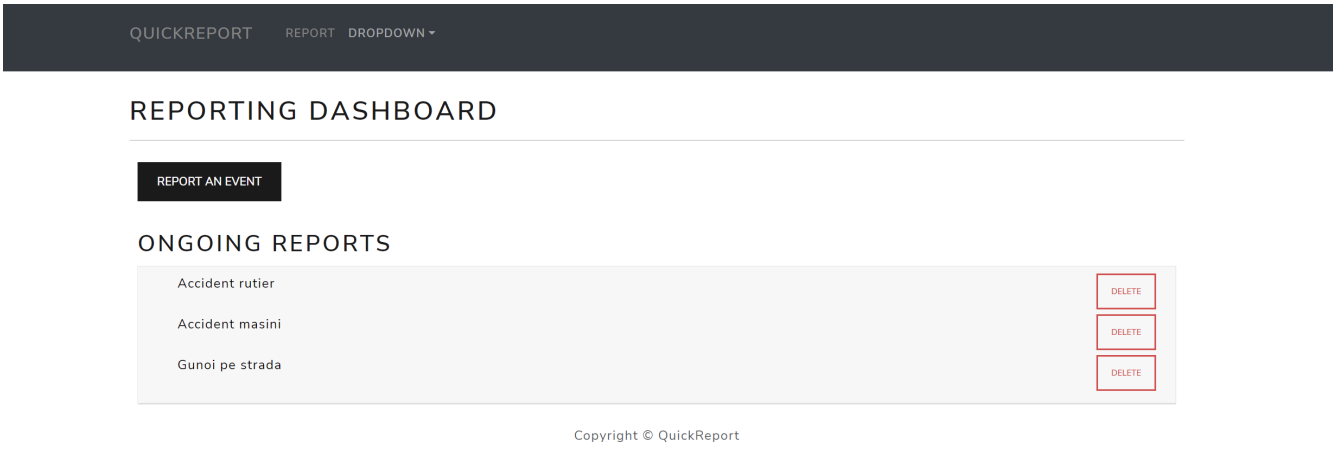

Figure 6. Dashboard

Figure 7. Insert Event

Implementation

In terms of design and implementation decisions, the project followed some key points of reference:

- AJAX approach - with use of axios library
- Minimal and Simple design with user focus
- Test driven development (TDD) - with use of SeleniumAPI
- Data Security - authentication based on jwt

Authentication

The Authentication process is straight-forward as the industry dictates and takes into consideration the security of the user. The password is encrypted using bcrypt and the whole process is based on the standard JWT that defines a compact and self-contained way for securely transmitting information between parties, the user receives an unique token for the current session, everything being incorporated into the Mongo database.

Insert Event Form

For the insert event form, it was used a form from React Bootstrap that creates a simple and reliable form for the users to report an event. The data is sent to backend on a post request from the backend that will create a new entry in the table Events. A relation with a user is required for an event.

Dashboard

The dashboard page looks different based on the type of the user logged into the application. If a user is a citizen, it will display the events created by him, with details about the event and the status. Frontend will make a HTTP request to the backend and this one will provide the frontend with the events. Also, a button is present that redirects the user to the insert event form. For a state authority user, the dashboard shows the events assigned to that authority. In database, each event has a category(salubritate, rutier, ordine, administrativ). To check for authority, the authority from the user's email is evaluated(e.g. stefan@administrativ.ro -> administrativ is the category). JSON objects are used for communication between frontend and backend.

Testing

In regards to testing, we went for an automated approach, working with SeleniumAPI to build a number of automated tests. The scope of the current suite is only focused on the authentication process and basic functionality, but the future plans take into consideration multiple suites to be integrated into a CI/CD pipeline for deployments.

Future Work

The first step for Quick Report to be used in production is to deploy the application on cloud. For this, team taught about deployment on AWS ECS, using Azure Pipelines for CI/CD. In this process, the automated tests will placed for a better user experience. The next step is to create new tests that will handle almost all the edge cases. For the insert event form, a new feature to add a picture or pictures is what the team taught can be improved. A survey will be conducted to improve the design and to add what users expected from this type of application.

References

1. Java tutorial. https://www.javatpoint.com/reactjs-tutorial. Accessed: 2022-05-21.

2. Nodejs intro. https://www.w3schools.com/nodejs/nodejs_intro.asp. Accessed: 2022-05-21.

3. Node.js. About. https://nodejs.org/en/about/. Accessed: 2022-05-21.

4. Node.js web application framework. https://expressjs.com/. Accessed: 2022-05-21.

5. The application data platform. https://www.mongodb.com/. Accessed: 2022-05-21.

6. SeleniumHQ. Seleniumhq/selenium: A browser automation framework and ecosystem. https://github.com/SeleniumHQ/selenium. Accessed: 2022-05-21.

Author contributions statement

O.P., and U.D. created the insert page form and dashboard, B.S. created the authentication and tests and S.P. wrote the article. All authors reviewed the manuscript and all authors took place in the design and preparation stage.