Final Year Project Report

Full Unit – Final Report

UTILISING HTML5 AND VECTOR MAPS TO CREATE AN OFFLINE MAP APPLICATION

Adil Mushtaq

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Reuben Rowe

Department of Computer Science
Royal Holloway, University of London

March 31, 2023

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 4590

Student Name: Adil Mushtaq

Date of Submission: 9/12/2022

Signature:

Table of Contents

Abstract

Web applications have been a core part of everyday life with functionality varying from word processing applications like Office 365, streaming services such as Spotify or Netflix, and social media applications like Pinterest or WhatsApp[1]. They're constantly evolving to ensure a high level of user experience across multiple devices with responsive design and consistent updates on all platforms. These applications combine the capabilities of their existing websites while having a simple yet recognisable interface making it easy to use on a multitude of different platforms such as mobile, tablet and browser. Many of these web applications can be described as a Progressive Web Application (PWA) which allows these apps to be installed to a device's homes screen/desktop without the need for a digital distribution system while also maintaining the user's security due to the requirement for HTTPS[6]. These applications also often have offline capabilities which allows them to be used even with a poor or no network connection. For example, Office 365 allows for local documents to be edited and then uploaded when an internet connection is found, and Spotify allows users to locally download their music at a specified quality to be listened to when offline. These features can also be seen in popular mapping applications like Google Maps which allows users to download map areas and directions, often as vector maps, for when the network connection is poor or not available which is common for user's travelling through rural areas or areas with low quality network infrastructure.

Chapter 1: **Introduction**

1.1 The Problem

The widespread use of mobile devices has made it easier than ever for people to access information and navigate their surroundings. However, many map applications require an internet connection to work, which can be a hindrance in certain situations, such as when traveling in remote areas or during a network outage.

An offline map application aims to solve the problem of not being able to access map data and navigate to a specific location when there is no internet connection. This can be an issue for people who rely on map applications for navigation, especially when they are in remote areas or during a network outage. An offline map application would allow these users to access the necessary map data and navigate to their destination even without an internet connection. This can be particularly useful for travellers, hikers, and other outdoor enthusiasts who may not always have access to a reliable internet connection.

In addition to this, an offline map application can provide several other benefits. For example, it can save users data usage and reduce their dependence on a reliable internet connection. This can be especially important for people who are traveling abroad and may be subject to high data roaming charges. An offline map application can also provide faster and more reliable access to map data since it does not have to be downloaded from the internet. This can be especially useful for users who are in a hurry or in an emergency. Overall, an offline map application can provide a more convenient and reliable way for users to access map data and navigate to their destination, even when they do not have an internet connection.

To develop an effective offline map application, I must consider several distinct factors. For example, I must ensure that the application is easy to use and provides the necessary features and functionality for navigation. I must also consider the size and format of the map data and ensure that it can be stored and accessed efficiently on the user's mobile device. In addition, I must consider the user's context, such as their location, their destination, and their mode of transportation, and ensure that the application provides the appropriate information and guidance for each situation. Finally, I must consider the user's preferences and needs, and ensure that the application is customizable and can be tailored to the individual user's requirements. By addressing these and other factors, I can create an offline map application that is effective and useful for a wide range of users.

One additional factor that I should consider when designing and developing an offline map application is the ability to update the map data. Since the map data is stored on the user's device, it may become out of date over time, as roads, buildings, and other features may be added, removed, or changed. To address this issue, the application should provide a way for users to update the map data periodically, either by downloading updates from the internet or by transferring the data from another source, such as a computer or another mobile device. This can help ensure that the map data remains accurate and up-to-date and can provide users with the most current information for their navigation needs.

1.2 Aims and Objectives

In this project, I will use HTML5, JS, CSS and OpenStreetMap (OSM) to develop a browser-based, vector maps application with offline capabilities. To do this, I will use the advanced features of HTML5 combined with JS and CSS to develop a usable interface for the application.

I will initially have to revisit standard HTML and CSS to relearn the basics needed to create a simple web page before moving onto the features of HTML5 which will allow me to create canvas objects. Recent updates in HTML5 also allow for further support for mobile browsers so that users can also use the app on the move in a lightweight package. The utility of HTML5 canvases would allow me to directly process and display the vector data required to generate the image seen by the user.

Alongside this, I will need to revisit JavaScript to develop the client-side scripts to allow users to interact with the map and, as an extension, search for POIs both online and offline. The data I will use for this will be gathered using the OSM database which will then be stored locally using an XML format.

Another key aspect of the project is the ability to work offline which would rely on application caching. However, after further research and a conversation with my supervisor, it was made apparent that this was a deprecated method of doing such and so I should focus on using service workers which check for availability of resources in local storage and retrieves them when available. This would be in collaboration with an indexedDB which will store the data from OSM.

Finally, as an extension to the project, I will attempt to develop a method for the user to dynamically download data from OSM when there is an internet connection available. This will be a toggleable feature which will automatically download data for tiles within a user specified radius of their current location when they are connected to the internet. This will provide a level of reliability for the user as they will always have local map information but also ensures that it does not overwhelm local storage.

1.3 Deliverables

For this project, it will be split into two parts. The first part will focus on creating a solid foundation in the technologies involved in the project with a proof-of-concept program being developed with a corresponding report on the information and techniques learnt.

1.3.1 A "Hello World" Offline HTML5 Application

This first proof of concept application focuses on consolidating my knowledge of HTML, CSS, and JS to create a simple hello world application which displays the web page title, the network status of the page and an image. It will also be able to be accessed offline. This will be paired with a report containing the methods and other technologies I used to create the application.

1.3.2 A "To-Do List" Application

The second focuses on utilising indexedDB to create an application in which the user can create one or many to-do lists which can then be stored in the database before the application is closed and then reopened with the same data stored. This would be paired with a report discussing the format of the database as well as any required security features.

1.3.3 Drawing Shapes using the HTML5 Canvas

HTML Canvas elements would be the next technology which I would research canvas elements and the way in which the objects are created experimenting with glyphs.

1.3.4 Gathering OSM Data

This application would utilise the OSM API to collect data from OSM and then display the raw data received with the report discussing the format in which it is received and how this could be used to render an actual map for the final product.

1.3.5 Final Product

As per the project specification, the final product should be able to load and display map data, interact with the map including the ability to zoom and move the map without sacrificing the map's visual quality. It should also work offline and have features to improve performance while rendering map data by utilising database indexing and image caching.

However, if time permits, the following features may also be added:

- Login/registration system
- Search for POIs
- Save POIs to saved locations or other user lists
- Dynamically download map data of a set radius of the user when a connection is present

Chapter 2: **Web Applications**

2.1 Web design

One of the key challenges in developing offline mapping applications is creating a user-friendly interface that allows users to easily access and interact with the map data. This is where web design plays a crucial role. By using responsive web design techniques, I can ensure that the application functions at a similar level on a range of devices, from smartphones to tablets to laptops.

In addition, a visually appealing design can improve the user experience and make the applications more enjoyable to use. For example, the use of vector map tiles and vibrant colours can make the maps more engaging and easier to read. The use of intuitive navigation controls, such as zoom and pan buttons, can also make it easier for users to explore the map data.

The usage of web-based mapping APIs, such as the OpenStreetMap API being used in this project, would allow me to incorporate map data into their applications and provide a range of features, such as the ability to search for specific locations, plot routes, and display information about points of interest.

2.2 Frameworks

Many offline mapping applications are built using web frameworks, such as ExpressJS, to make it easier and faster to develop complex, interactive applications. ExpressJS is a web framework that is commonly used for building web applications, including offline mapping applications. It is a fast, minimalist framework that is built on top of the Node.js runtime environment and is designed to make it easy to create web applications that are scalable, efficient, and reliable.

One of the key features of ExpressJS is its routing system, which allows developers to specify the URLs that the application will respond to and the corresponding actions that should be taken when those URLs are requested. This makes it easy to create complex, multi-page applications with a clear and consistent structure.

Another important feature of ExpressJS is its middleware system, which allows developers to add custom functionality to the application at various stages of the request-response cycle. This enables developers to easily add features such as authentication, data validation, and error handling without having to write complex code.

One of the key benefits of using ExpressJS for building offline mapping applications is its support for server-side rendering. This means that the application can generate the HTML for a web page on the server, rather than relying on the client-side JavaScript to render the page. This can be especially useful for offline mapping applications, as it enables the application to provide map data and other content to the user even when the device is not connected to the internet.

In addition, ExpressJS provides support for caching and other performance optimization techniques that can help offline mapping applications run smoothly and efficiently even when the device is offline. For example, the application can cache frequently used data and resources on the user's device, allowing the application to access the data quickly without having to fetch it from the server.

2.3 Progressive Web Applications

There are now examples of offline mapping applications are now being developed as PWAs, which use modern web technologies to provide users with a native app-like experience. One of which being the popular offline mapping app HERE WeGo is available as a PWA, which allows users to access the app and use its offline mapping functionality even when they are not connected to the internet.

Furthermore, they're built using web technologies such as HTML, CSS, and JavaScript, which are the same technologies used to create traditional web applications which makes it much simpler to develop one. However, PWAs use additional features and APIs provided by modern web browsers, such as service workers and the web manifest, to provide a more app-like experience.

The utility of service workers as well as a web manifest also extends to providing offline functionality, allowing users to access the app and use its features even when they are not connected to the internet. This can be particularly useful in situations where network connectivity is poor or non-existent, such as when traveling in remote areas or on an airplane.

PWAs can also be installed on the user's device, allowing them to launch the app from their device's home screen and access native device features such as the camera or GPS. This can make the offline mapping application more convenient and user-friendly and can improve user engagement and retention.

In addition to providing offline functionality and a more app-like experience, PWAs can also improve the performance and reliability of offline mapping applications. For example, PWAs can use caching and other techniques to reduce the amount of data that needs to be downloaded over the network, which can improve the app's performance and reduce the impact of network outages.

Chapter 3: **Software Engineering**

3.1 Methodology

For this project, I used the agile software development methodology to guide the development process. Agile is a flexible, iterative approach to software development that emphasizes continuous feedback, and frequent iteration. Thus, due to the 3 meetings with my supervisor, with potential for more, per term, I would be able to split the project into approximately 8 sprints of around 2/3 weeks.

With agile, I would be able to break down the project into smaller pieces and focus on completing tasks quickly within the sprint making the whole project much more manageable as I knew exactly what I had to do between meetings.

3.2 Testing

In this project, I will utilize test-driven development (TDD) to ensure the quality and correctness of my code. Using TDD, I will write tests for each feature of the application before writing the code itself. This will allow me to catch any errors early in the development process and ensure that the code meets the requirements of the project.

For example, when implementing the feature that allows users to save maps for offline use, I will first write a test that defines the expected input and output for the save function. The test will specify that the function should accept a map and a location as input and should save the map data to the device's local storage.

Next, I will write the code for the save function, making sure that it passes the test I have written. If the code doesn't pass the test, I will go back and modify the code until it does. This process will help me to ensure that the save function is correct and works as expected.

Using TDD will allow me to develop the application with confidence and ensure the quality of the code. It will also help me to catch any errors early in the development process, which will save me time and effort overall.

3.3 Documentation

To help others understand and use the application, I will create a variety of documentation to provide information about the project and its features. This will include a project overview, a description of the application's features and technical implementation, installation instructions, user demonstrations, and code documentation.

The project overview will provide an overview of the purpose and goals of the project, as well as any relevant background information. The description of the application's features and technical implementation will provide details on how the distinct parts of the application will work and interact with each other, as well as the technologies and tools I will use to develop the application. The installation instructions and user guides will provide step-by-step instructions for setting up and using the application. Finally, I will write JSDoc comments inside my code so that I compile this and create a suitable code documentation.

Chapter 4: **Program Development**

4.1 Hello World Offline Application

This POC was created using a tutorial by Log Rocket[9]

4.1.1 Project Structure

The project structure was initially as follows.

Directory	Purpose
hello-world	Act as main project directory
/certs	Contain SSL files
/node_modules	Contain files made during dependency installation
/public	Act as entry point for application
/public/images	Contain images required by application Contain icons needed for conversion to PWA
/public/js	Contain JS files

4.1.2 Designing the Web Page

The content of the web page is shown in index.html. Here are the parts of the code relevant to this stage of development.

```
<head>
    <title>Hello World</title>
    <link rel="stylesheet" href="style.css">
    <h1 id="pagetitle">Offline 'Hello World' web application</h1>
</head>
<body>
    <span>You are now</span> <span><b class="page-
status">online</b></span>
    <p id="imgcaption">Here's an image</p>
    <img id="globe" src="images/globe-512.png" alt="Globe image">
    <script src="/js/status.js"></script>
</body>
```

This code creates a webpage which has the title 'Hello World' and the main heading giving the name of the application. It also applies the styling from style.css shown below.

```
body {
    background-color: azure;
```

```
}

#pagetitle {
    font-size: 50px;
}

p {
    color: black;
}

#globe {
    border: 10px solid;
    padding: 15px;
}
```

The body of the web page displays the network status of the application alongside a bordered image. The network status is changed by the status.js file which changes the text if the network is offline.

4.1.3 Developing Locally Ran Application

The next stage of the project involved implementing the functionality of the application and creating a local server for the application. This began with generating the package.json file by running npm init in the command line and then filling out the questions. Both are shown below.

```
package name: (report-2-to-do-list)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
```

```
{
  "name": "hello-world",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "server-debug": "nodemon --inspect server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "nodemon": "^2.0.20"
  },
  "devDependencies": {
    "jsdoc": "^4.0.0"
  }
}
```

ExpressJS was then installed as the HTTP server with Nodemon as the debugger using `npm install express nodemon` in the command line. However, to set up Nodemon correctly, an additional line `"server-debug": "nodemon --inspect server.js"` was added to the package.json file (above). The server.js file (below) acts as the main server for the application and deals with the requests with the port 80 sent to the localhost address. The lines 10-12 directs the user to the main page when they type in localhost in their browser:

```
const express = require('express')
const path = require('path')

const httpPort = 80

const app = express()

app.use(express.static(path.join(__dirname, 'public')))

app.get('/', function(req, res) {
  res.sendFile(path.join(__dirname, 'public/index.html'))
})

app.listen(httpPort, function () {
  console.log(`Listening on port ${httpPort}!`)
})
```

Now, when the user was to run npm start in the command line, the user would be able to see the main page when they go to localhost in their browser. If they were to run the Lighthouse Audit on this application, they would see a similar result to this (below).

As seen, they are plenty of criteria that are not satisfied for the application to be classified as a PWA.

4.1.4 Completing Required Steps for a PWA

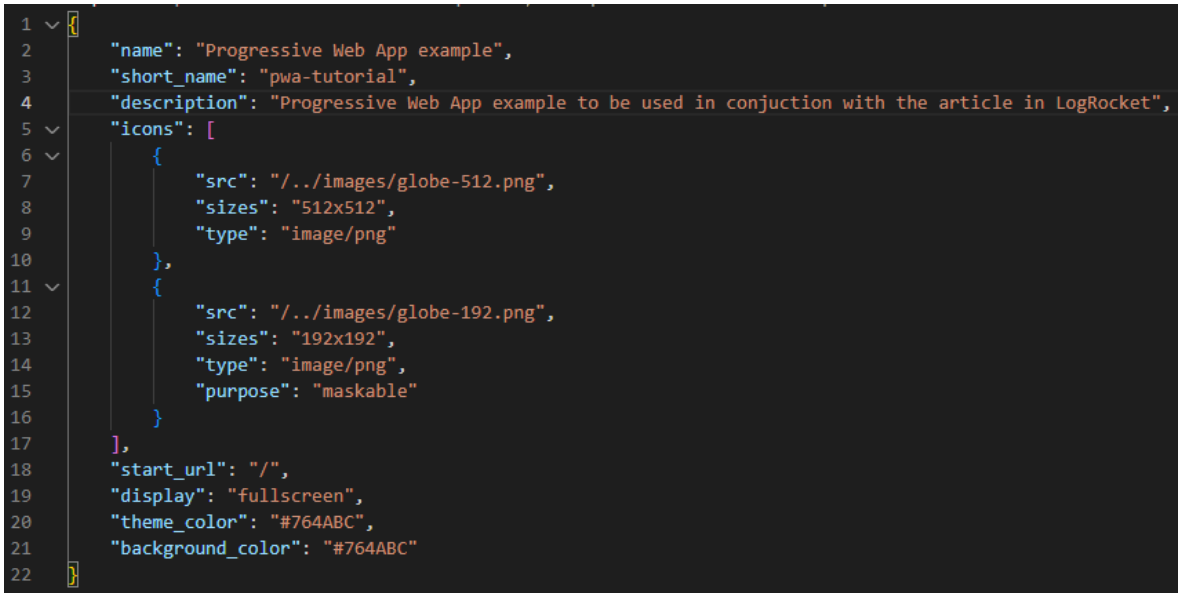
To satisfy these criteria, it is easy to group them by what they require. For example, under the PWA optimisation subheading, the following are seen:

- Does not set a theme colour for the address bar.
- Content is not sized correctly for the viewport.
- Does not have a `<meta name="viewport">` tag with `width` or `initial-scale`.

These can all be satisfied by making changes within the index.html file and adding the following lines which set the theme colour and scale the content for the device being used.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="theme-color" content="#764ABC">
```

The next set of criteria focus on the implementation of a web manifest which stores the details required for the installation of the PWA on a device. This includes: the name of the app, a short name (optional), description, any icons which will be used to display the app on the home screen/desktop of the device the app is installed on, the start_url, the type of display the app will take, and the global theme/background colour. This file should look something like this (below).



```
{
    "name": "Progressive Web App example",
    "short_name": "pwa-tutorial",
    "description": "Progressive Web App example to be used in conjuction with the article in LogRocket",
    "icons": [
        {
            "src": "/../images/globe-512.png",
            "sizes": "512x512",
            "type": "image/png"
        },
        {
            "src": "/../images/globe-192.png",
            "sizes": "192x192",
            "type": "image/png",
            "purpose": "maskable"
        }
    ],
    "start_url": "/",
    "display": "fullscreen",
    "theme_color": "#764ABC",
    "background_color": "#764ABC"
}
```

This file would then have to be linked within the index.html file with the line <link rel="manifest "href="js/pwa.webmanifest"> to apply these changes. They would then have to add an additional line to the index.html file to declare the add-to-home-screen (AHS)/apple touch icon.

The application would then have to be served by HTTPS to be validated as a PWA as well as to register a service worker (SW) which allows the application to work offline. To do this, a self-signed certificate would have to be created as well as ensuring that all requests to the application would always be over HTTPS instead of HTTP. To do this, the following commands would have to be ran in an administrator-approved command line.

```
// Create RSA-2048 key with password
openssl genrsa -des3 -out rootCA.key 2048

// Create root certificate (Requires optional information)
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1460 -out
rootCA.pem

// Trust certificate locally
certutil -addstore -f "ROOT" rootCA.pem
```

After this is done, an OpenSSL configuration file and a v3.ext file need to be made in order to run the following code:

```
// Create a private key and certificate-signing request (CSR) for the
localhost certificate.
openssl req -new -sha256 -nodes -out server.csr -newkey rsa:2048 -keyout
server.key -config server.csr.cnf

// Issues a certificate via the root SSL certificate and the CSR
openssl x509 -req -in server.csr -CA rootCA.pem -CAkey rootCA.key -
CAcreateserial -out server.crt -days 500 -sha256 -extfile v3.ext
```

The server.js file would then have to be updated to reflect this. A new file pwa.js would then finally be able to register the SW by adding implementing this function:

```
function init() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/sw.js')
      .then((reg) => {
        console.log('Service worker registered -->', reg);
      }, (err) => {
        console.error('Service worker not registered -->', err);
      });
  }
}
```

You would then have to assign this script to index.html by adding another <script> element: <script src="js/pwa.js"></script>. The last step is to write the code for the SW which includes assigning files to be added to the cache, registering the SW, and adding the files to the cache, updating the cache every time the website is accessed while online, fetching requests from cache or network depending on the user's connection.

4.1.5 Deploying the Web Application

At this point the application is ready to be tested a final time. However, if the server is run and tested on the browser, the SW would likely come up with a registration error as the certificate is self-signed which means it would not be trusted by most web browsers. To avoid this Chrome would have to be ran with the following command:

```
chrome.exe --ignore-certificate-errors --unsafely-treat-insecure-origin-as-secure=https://localhost:443
```

This command runs Chrome in a way in which it will ignore this error and register the SW as normal.

4.2 Final Product

4.2.1 Core Features

As per the project specification, the final product should be able to:

- Load and display map data
- Interact with the map including the ability to zoom and move the map without sacrificing the map's visual quality
- It should also work offline
- Have features to improve performance while rendering map data by utilising database indexing and image caching.

4.2.2 Additional Features

Other features that I will attempt to add to the final product include:

- Login/registration system
- Search for POIs
- Save POIs to saved locations or other user lists

- Dynamically download map data of a set radius of the user when a connection is present

4.2.3 User stories

From these features, I have prepared a list of user stories:

- As a user, I should be able to register for an account.
- As a user, I should be able to login into my account.
- As a user, I should be able to view a comprehendible map with road names and suitable place names.
- As a user, I should be able to view cached map data when no network connection is present.
- As a user, I should be able to move around the map.
- As a user, I should be able to zoom into/out of the map without losing quality.
- As a user, I should be able to search for an address or place.
- As a user, I should be able to get directions from one address/place to another.
- As a user, I should be able to create collections of addresses/places.
- As a user, I should be able to save locations/addresses into a collection.
- As a user, I should be able to remove entries from a collection.
- As a user, I should be able to rename collections.
- As a user, I should be able to toggle dynamic download of map data.
- As a user, I should be able to change the radius of the user from which this map data is downloaded.

Chapter 5: **Evaluation**

5.1 Timeline

The project was initially laid out in the following set of milestones.

Term one:

Week 1. (w/c 26/09/22) Research basis of project.
Week 2. (w/c 04/10/22) Research differences in mapping technologies. Submit project plan.
Week 3. (w/c 11/10/22) Refresh knowledge on HTML, JavaScript, CSS, and research how to utilise HTML5 and canvas objects to create a demo offline application.
Week 4. (w/c 18/10/22) Consolidate research from previous week in a report.
Week 5. (w/c 25/10/22) Research service workers, local storage and indexedDB. Start work on a to-do list application. Supervisor meeting.
Week 6. (w/c 31/10/22) Complete previous weeks application and consolidate research in a report.
Week 7. (w/c 07/11/22) Research how to process data received from OSM and possible data structures that would be compatible with format in which data is received.
Week 8. (w/c 14/11/22) Consolidate research from previous week in a report.
Week 9. (w/c 21/11/22) Start work on interim report and finish off any remaining research. Start presentation. Supervisor meeting.
Week 10. (w/c 29/11/22) Finish and submit interim report. Complete presentation.
Week 11. (w/c 05/12/22) Presentation.

Term two:

Week 1. (w/c 09/01/23) Begin work on front-end design and create template for the application.
Week 2. (w/c 16/01/23) Continued.
Week 3. (w/c 23/01/23) Design settings page where user can adjust preferences within the application.
Week 4. (w/c 30/01/23) Start implementing methods of collecting and displaying collected data from the OSM DB.
Week 5. (w/c 06/02/23) Continued.
Week 6. (w/c 13/02/23) Start work on storing the data collected locally to be used offline.
Week 7. (w/c 20/02/23) Continued.
Week 8. (w/c 27/02/23) Test program under load and check for possible optimisations.
Week 9. (w/c 06/03/23) Incorporate search features and login system. Start implementation of dynamically downloading data when connected to the internet.
Week 10. (w/c 13/03/23) Continued. Final testing.
Week 11. (w/c 20/03/23) Submit final report.

5.2 Current Progress

As of the time of this report I have completed all milestones from week 1 – 4 and completed the presentation.

5.3 Next steps

Therefore, the next steps to be taken for my project involve:

- Completing the rest of the POC programs and reports by the end of the Christmas break.
- Begin work on development of my final product using the timeline for term 2.

Bibliography

[1] *10 essential examples of web applications* (2022). Available at: https://www.tutorialsmate.com/2022/04/examples-of-web-applications.html (Accessed: December 8, 2022).

[2] Bertolotto, M. and Egenhofer, M.J. (2001) "Progressive Transmission of Vector Map Data over the World Wide Web," *GeoInformatica*, 5(4), pp. 345–373. Available at: https://doi.org/10.1023/a:1012745819426.

[3] Galov, N. (2022) "17+ Google Maps Statistics to Survey in 2022," *Web Tribunal*, 6 April. Available at: https://webtribunal.net/blog/google-map-statistics/#gref (Accessed: December 8, 2022).

[4] *Introduction to progressive web apps* (2022). Available at: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction (Accessed: December 9, 2022).

[5] Issac, A. (2020) *How to create trusted self-signed SSL certificates and local domains for testing, Medium*. Better Programming. Available at: https://betterprogramming.pub/trusted-self-signed-certificate-and-local-domains-for-testing-7c6e6e3f9548 (Accessed: December 9, 2022).

[6] Kitakabee (2022) *Are progressive web apps (PWA) The Future of Web Applications?, BrowserStack*. Available at: https://www.browserstack.com/guide/future-of-progressive-web-apps (Accessed: December 8, 2022).

[7] Miller, C.A. *et al.* (2012) "SCRIBL: An HTML5 canvas-based graphics library for visualizing genomic data over the web," *Bioinformatics*, 29(3), pp. 381–383. Available at: https://doi.org/10.1093/bioinformatics/bts677.

[8] Mountjoy, D.N. *et al.* (2000) "Basing non-linear displays on vector map formats," *Journal of Navigation*, 53(1), pp. 68–78. Available at: https://doi.org/10.1017/s0373463399008620.

[9] Shah, D. (2021) *How to get HTTPS working on your local development environment in 5 minutes, freeCodeCamp.org*. freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/how-to-get-https-working-on-your-local-development-environment-in-5-minutes-7af615770eec/ (Accessed: December 9, 2022).

[10] Spínola, D. (2020) "How to build a progressive web app (PWA) with Node.js," *Log Rocket*, 6 April. Available at: https://blog.logrocket.com/how-to-build-a-progressive-web-app-pwa-with-node-js/ (Accessed: December 9, 2022).