Workneh Tefera Tamire

HTML5 and Its Capability to Develop Offline Web Applications

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

22 April 2016

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Workneh Tefera Tamire HTML5 and Its Capability to Develop Offline Web Applications 42 pages 22 April 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Olli Hämälinen, Senior Lecturer

The main purpose of this bachelor's thesis project was to study the capabilities of HTML5 to develop offline web applications. Building a web application using application cache and local storage and demonstrating a prototype that works without any network connection was the goal of the study.

The literature review part of the study assesses different academic studies on the possibility of developing an offline web application with a special emphasis on HTML5's application cache and local storage. The frontend of the prototype website was developed using HTML5, CSS and JavaScript and the backend was built using the PHP and MySQL databases.

A demo web page was built and tested on different client browsers. The outcome of the test was that the web application performs offline without any network connection and the planned goal of the project was achieved.

To summarize, the application solves erratic internet connection problems, loads fast and reduces server load. Nevertheless, the field requires further study in order to improve user experience while working offline. Hence, it is recommended that future students conduct broadened research on HTML5's offline functionality with great consideration on the mechanism to increase the storage capacity of local storage.

Keywords	Capability of HTML5, offline web application, application cache, local storage

Contents

Appendices

Abbreviations and Terms

API	Application Programming Interface
CDN	Content Delivery Network
CSS	Cascading Style Sheets
CPU	Central Processing Unit
DOM	Document Object Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Secured Hyper Text Transfer Protocol
IndexedDB	Indexed Database
MIME	Multipurpose Internet Mail Extensions
2D	Two-dimensional
3D	Three-dimensional
SQL	Structured Query Language
UI	User Interface
UX	User Experience
W3C	World Wide Web Consortium
WWW	World Wide Web

1 Introduction

These days, the importance of web applications as compared to desktop applications is increasing to a large extent. Flexibility, platform independence and easiness to deploy, make the web application to be the primary option and answer most of the key constraints of traditional applications. It reaches to the viewers with the least effort required due to cross-platform compatibility. Unlike their counterpart, web applications are not configured or need to be installed on the computer of the user before their use.

Most web browsers have been able to cache some of the static resources on the user's device since developers had no way of checking what they wanted to be cached. As a result, web applications currently use JavaScript to accomplish this task. JavaScript is one of the widespread client side scripting languages and web designers include a lot of JavaScript so as to see the load that occurs on the web server due to the flow of data between the web server and the client device.

One method to reduce the web server burden is to minimize the number of requests browsers send to it. Furthermore, a lower request to the web server means improved service by web applications. Hence, one way to achieve this is to save important information or some portion of the web pages that are vital to run the application on the user's machine. Moreover, this traffic will be further reduced if clients use web applications offline.

Nevertheless, building web applications that work without a network connection requires cautious design concerns. In addition, developing and maintaining a rich web application that has an offline-enabled functionality is challenging. However, with recent technological advancements in Hyper Text Markup Language (HTML), building applications that run without an internet connection has become possible.

The main purpose of this project is to develop an offline web application using HTML5. The application is planned to be tested and assumed to improve the users' experience.

2 Literature Review

2.1 Overview of Web Applications

Initially, establishing a platform for data distribution and displaying information to everyone over the world was the plan of the World Wide Web (WWW) [1,100]. The main motive was to discover a system that enables to pass reliable information internationally using it as an interface. Moreover, the source of the content to be used for this web would come from all over the globe. As a result, the architectural setup requires a network connection for effectiveness and efficiency. Nevertheless, with the improvement and growth of technologies and expansion in business necessities, motivated web applications to grow into multifaceted applications [2,11].

Furthermore, web applications nowadays are a popular choice as compared to the old day desktop applications due to the fact that they are simple to deploy and maintain. Hence, the architecture of web applications, unlike the desktop application depicts that they can be easily deployed on a central web server instead of each machine that entails the application. As a result, using this model enables businesses to allocate the application easily to a large number of users [2,13]. Figure 1 shows different components of online web applications.

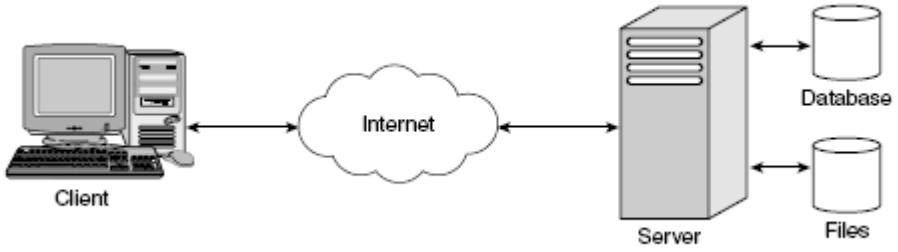

Figure 1. Core elements of online web applications. Reprinted from Webreference [3].

As illustrated in Figure 1 above, web applications perform better when the architectural environment of the application has an internet connection. Therefore, with the availability of connection, the user works together with the web page by requesting to the remote web server which in turn responds to the client for the application to perform as demanded [3].

However, this method of submitting and receiving data can be a big problem in cases of low internet coverage or high latency. Due to this, web applications could not function as required since it wastes time waiting to get data from the server that finally end up with non-responsive interface. As a result, building a web application using HTML5's offline services minimize the problem and increase user experience [4].

2.2 Offline Applications

As explained in the previous section 2.1, applications work as desired when the architecture that surrounds a typical web application requires connectivity to the Internet. Therefore, users submit a request to the server to interact with the web page though there are round trips between the client and the server sequentially so that the web page performs as required. However, this can be a big problem in scenarios where the network is a major issue due to the fact that applications take a long time to get a response resulting in an unwanted, unresponsive or less reactive user interface [4].

The logic that follows offline web application is that it keeps on working in situations where there is no internet connection, unlike a standard web application. This kind of web application saves important web application resources and user data on the client machine persistently instead of bringing information from the server when requested. As a result, it lets programmers build high speed web applications since the application uses resources stored on the local device. Furthermore, it helps to raise the application rate when fetching data as the information is stored locally reducing the standard round trips [4].

Application's capability to operate offline is valuable primarily when there is no access to internet connection but it is a must for the application to work. Moreover, offline application strives and pays attention more to the required information though there are certain deferrals within the structure. Hence, in order to make an offline application that performs successfully, it is vital and recommended to check, update and make secure the resources saved at the user device.[4.] Figure 2 below depicts core elements of offline web applications.

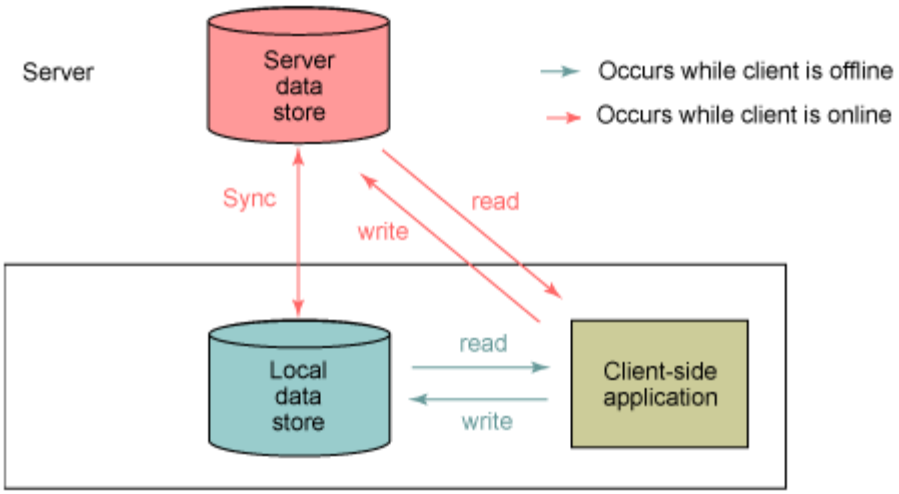

Figure 2. Core elements of offline web applications.
Copied from IBM DevelpersWorks [5]

As shown in Figure 2 above, offline web application uses locally saved resources when there is no connection to the network. On the other hand, when there is connection to the internet, data can be read or write directly from the server [5].

However, building web applications that work offline requires cautious design concerns. Developing and maintaining a rich web application that has an offline enabled functionality is challenging. As conventional web applications perform first by sending requests from the client and then receiving responses from the server, many business owners are investing their time and effort to make certain that these servers stabilize the burden and shrink the usage of internet bandwidth [6,121].

As a result, due to the current advancement in web principles, HTML5 offers a variety of applications that enable websites to perform with files saved in the client machine. Hence, due to these, applications can take full advantage of the power set by the client machine to increase image size rather than uploading large file at the server. [7,272.]

2.3 Document Object Model

One way to integrate web application with a user device is using the Document Object Model which is known as the DOM. This technology enables scripting languages like JavaScript to discover a range of features that acknowledge users to communicate with applications to be implemented. Furthermore, different elements of web applications are included in the client are using the DOM. As a result, DOM is the boundary that delimits local storage and allows accessing a user's device. Therefore, this is the main cause for the database interface to build up in JavaScript. Moreover, this scripting language runs on the browser that has permission to use the DOM and the architecture can be shared by all local storage schemes [8,132].

Before building applications using HTML5, one needs to know the structural design that HTML5 shares. It gives an interface called the Application Programming Interface (API) which gives permission to cache web pages with or without browsing them. Furthermore, a client's move web contents after viewing the web pages for the first time. [9, 1002.]

However, doing so requires a web builder to search for possible options to determine the application used in case it is important to transfer a portion of the resources. Furthermore, when users access the web page, HTML5 has a facility to check the availability of network and detect being online or offline [9,1007]. The HTML DOM tree is shown in Figure 3 as follows:

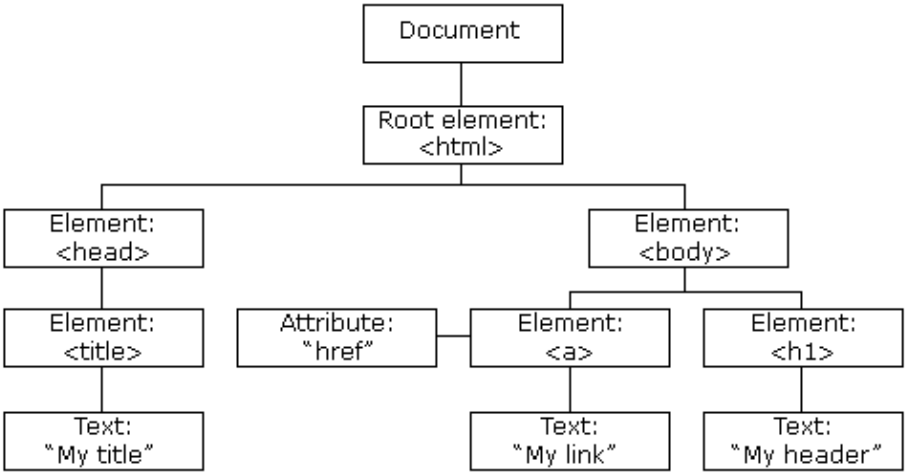

Figure 3. The HTML DOM Tree. Reprinted from W3schools [10]

As demonstrated in Figure 3, the HTML DOM tree contains elements, attributes and texts. The root element is divided broadly into the head and body elements each further comprises different elements, attributes and texts [10].

2.4 Web Caching

When using a web application or browsing information from the web, two things are very important: speed and efficiency. Everyone is satisfied while working with high-speed applications and service providers also strive to get the largest part out of the existing bandwidth. Moreover, when designing web applications, caching requirements enable to improve the required information access time since it reduces the network traffic [11]. In contrast, no one is willing to work with low bandwidth websites that requires greater patience [12].

As a result, web caching enables to raise the performance and scalability of applications since it is a method that optimizes the speed of an application. Furthermore, cache allows saving copies of the web application pages on the users machine in order to access later when there is no connection. [13,538.]

In addition, storing copies of Cascading Style Sheet (CSS), JavaScript, Image, HTML and other resources in the web browser reduces the time required to access the data again from the web server. On the other hand, when request is made to a web server, many resources are sent back to the user and requesting the server for the same page for the second time means making the web server busy which results in high web traffic. [15, 538-540.]

There are two types of web cache: public and private. The first cache is shared by many users whereas the latter is used only by one user. In case of public cache, there is a proxy server that cache resources which are frequently used by a number of users. On contrast, the browser stores private cache in the user's device so as to enable the user access immediately instead of retrieving from the server [15].

2.5 Performance and Scalability

2.5.1 Performance

There are a number of ways to measure the performance of a web application. Some of the web performance rules are listed as follows:

- Reducing number of HTTP requests from the web browser to the server.
- Enabling the server to send as little information as possible to improve the speed of web page loading.
- Sending information to the web page occasionally to increase user experience.

Moreover, response time to user interactions, throughput, and resources used for a given workload are among some. Page responsiveness measure is based on the users' view while the other two are measured using the whole system considering every end user [14].

2.5.2 Scalability

Scalability is the capability of a web application to handle the extra load while retaining an equivalent level of performance by adding further resources to the server. These resources can be hard disk storage, memory, Central Processing Unit (CPU) or other servers to the existing web environment.[14.]

2.6 Hypertext Markup Language and Offline Web Applications

HTML is a markup language used to describe the look and appearance of documents for making the web. HTML is undertaking continuous development since the beginning of 1990. Its latest version, HTML5, was released as a stable version of World Wide Web Consortium (W3C) on 28 October 2014. Moreover, with recent development in technology, W3C planned to improve the language specification recommendation for HTML5.1 that will be targeted at the end of 2016. [15,4.]

The gap between web and desktop applications is more and more narrowing by HTML5 and will probably bridge it entirely. HTML5 is among those recent web technologies that have been broadly acknowledged by Web developers. It provides features like dynamism, responsiveness, and faultless performance through the different browsers and mediums with lots of prominent features [15,12].

Among new elements that come with HTML5 are HTML UI controls, local storage support, multi-media support, two-dimensional (2D) and three-dimensional (3D) graphics and many other interesting new application features. As a result, they can be used to make web applications that are interactive and perform better with or without requiring customization. Moreover, these applications can be installed on mobile and desktop devices [15, 13].

In addition to some of the features mentioned above, HTML5 has new features that allow applications to work offline in cases where network availability is limited or no connection. Hence, in order to develop an offline web application, HTML5 allows application caching and offline storage. Some of the new storage APIs also called HTML5 storages that enable to create an offline web application is local storage also called DOM storage or web storage. Other storages that also help to create an offline web application are: Structured Query Language (SQL) Database, Indexed Database (IndexedDB) and File API. [15.]

Most of the above mentioned storages share the principle of same origin which means data storage is fixed to the domain it creates. Therefore, the data cannot be retrieved by any other origin instead by the site that creates it. As a result, in all of these storages, one to many store and origin principle applies where the latter is the number of different storage ways applied by the client browser. [15.] In the following part of this study, different technologies that enable to develop offline web applications are discussed in detail.

2.6.1 Local Storage

Local storage APIs were introduced by HTML5 to offer local data storage for offline web applications and eliminate the size constraints that exist in cookies [16]. Local storage also called Web storage stores data in key-value pairs and web application accesses this data using JavaScript. Moreover, unlike cookies that are less secure, this API provides better security and has greater privacy features. Furthermore, it is easier and uses less JavaScript syntax to set and get data as compared to cookies [17].

Web storage provides greater space to store data in contrast to using cookies which have less storage space. In addition, web storage minimizes overhead since the data stored on it does not require being part of every request. Also, web storage stores the data on the client browser and this data is accessed when request is made by the client and web server does not send every time whereas cookies are transmitted with each request [18].

Web storage has two different essences: local or session storage. With HTML5, data stored in the first one is persistent and permits users to use after the request is made. This means that closing the browser or restarting the computer has no impact on the data stored in the local storage and it will still be accessible [19].

As a result, this type of web storage is important in cases where the stored data exist across different windows and persist after the current session. Hence, local storage is called persistent storage and provides space for domains constantly. Moreover, it saves the data used between different sessions and closing and reopening a browser has no impact on it. [15,5.]

Concerning session storage, it is similar to the local storage in the API and only differs from the local storage in that it retains the data till the browser is closed. Therefore, a new window or tab opened after the session is closed will start a new session. This type of storage is for instance important in banking operations since the data are sensitive and will not be used after the browser is closed.[15, 6.]

2.6.2 IndexedDB

IndexedDB is local browser database developed by Oracle in the year 2009. It is a new API and has the ability to provide data storage in client browser by creating a platform independent mechanism for data caching. Furthermore, it creates an interface that allows to store and access objects in a key-value pair and provides sequential traversal of keys. To implement IndexedDB, B-Tree data structures are used since they are good for the big amount of data insertion and deletion. [17.]

IndexedDB follows the file origin policy to read the data stored on the client and check that the resources saved at the client cannot be accessed elsewhere instead it is limited to one domain. Nonetheless, in contrast to web storage, its API permits the storage of big data that is structured at the client. Also, as its name indicates, IndexedDB uses indexes to carry out high-speed searches.[17.]

Key characteristics of IndexedDB:

- It is an object-oriented database.
- It saves data as key-value pairs.
- It uses queries on an index and does not use SQL.
- It reads data only from the same web domain and stops reading from different origin.

At present, only some browsers support IndexedDB. Furthermore, both synchronous and asynchronous API accesses are allowed by IndexedDB. As stated above, all IndexedDB processes carry out within the context of the transaction giving a guarantee to the database activities to integrate between different requests [17].

2.6.3 Web SQL Database

Web SQL database is an offline SQL database and its implementation is based on SQLite database which is a general-purpose open-source structured language engine. Its primary feature is that it enables writing SQL statement on the client side [20].

Moreover, it comes with all its merits and demerits of the customary databases. It has a relational database makeup providing the user to use joins to manipulate and query data. Besides, it supports for transactions, protecting the database from isolation which is also witnessed in Web Storage [20].

However, there is a dependence created when using this query language. Due to this, there are different providers and the test of these SQL varies as suppliers, for instance, Oracle's SQL or Microsoft's Transaction SQL. It is complex and not compatible with many browsers and hence, at present Mozilla and Microsoft does not support it.[20.]

2.6.4 Application Cache

Most web browsers cache web pages immediately once they are visited by the user so as to offer a high-quality user experience. However, caching the web page automatically has some restrictions. The first restriction is that web developers lack access to make decisions on which web pages to cache in situations where there is no network connection. The second problem is that caching a website automatically is not a guarantee to build and access cached pages when there is no internet connection. [22.]

As a result, so as to solve the above mentioned problems and take care of them, HTML5 has offline web application API (App Cache). Using this technology, web builders can easily decide which resources to be cached and be available when the application is working offline. Hence, App cache enables the developer to make a decision in advance to choose what the planned website looks when it is offline. [22.]

Therefore, the HTML5 application cache is very critical in supporting offline web applications. As a result, using this technology enables the website to work properly in cases when there is no internet connection if implemented appropriately [22]. High level HTML5 architecture to build an offline application is depicted in Figure 4 below.

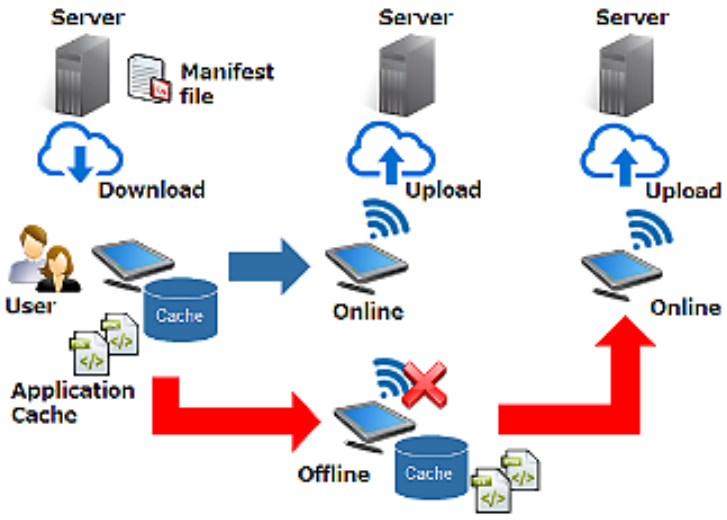

Figure 4. High level HTML5 Architecture. Copied from Micronet Techno [21]

As illustrated in Figure 4 above, when a request is made by accessing a single page, it enables to download all files that are listed under the applications manifest file. Once the specified files under manifest file are loaded, successive applications carry on using it with or without connection to the internet [17].

To summarize, different literatures related to the new features of HTML were reviewed and most of them wrote a lot about the importance of HTML5 to develop dynamic, interactive and user-friendly web applications. Furthermore, in addition to its ability to build an online web application, it also supports to develop an offline web application. Therefore, HTML5 has the capability to improve the user experience by offering different functionality to build an offline web application that performs without connection but gives the same function like online application [17].

3 Requirement Analysis and Designing the User Interface

3.1 Requirements for Offline Web Applications

In an offline web application, the main thing is to check whether the application is capable of working without an internet connection or not. Furthermore, the entire resources that are static and form the web page need to be saved on the user's machine so that they will be accessible from the client device without requiring connecting to the remote web server. Most of the time, HTML, CSS, JavaScript and image files are among the main files that constitute the web page. Doing so enables one to retrieve only those pages which are changed from the server and saved locally on the client's device [15].

Another essential requirement when building an offline web application is to store dynamic pages locally on the client device so that the web applications work offline. This is because web applications constitute not only static pages like HTML, CSS, JavaScript and images but also store dynamic pages that are created on a remote database. This dynamic data created on the remote database and saved locally need to be synchronized when there is an internet connection. [15]

After all important resources are stored, the next step is to check whether the application works without a connection or not. To check this, the general rule is to make sure the availability of an internet and then enable an online or offline mode based on the result. One way to check this is to use *window.navigator.onLine*. [15,11.] Listing 1 below shows a JavaScript code which enables one to check online status.

```
If (navigator.onLine) {
   alert ('application is online');
}  else  {
   alert ('application is offline);
}
```

Listing 1. Online status check

As illustrated in Listing 1, *window.navigator.online* alerts the user about the existence of a network. Hence, in order to get a notification about the status of change in the network, a subscription to a window event is needed as shown in Listing 2 below:

```
Window.addEventListener ("offline", function (e) {
   alert ("application is offline");
}, false);
window.addEventListener ("online", function (e) {
   alert ("application is online");
}, false);
```

Listing 2. Window event

As shown above in Listing 2, a notification message will be sent if the application is online or not. This alert message enables the user to access either the cached files if the message is *"application is offline*" or all files that are available online if the message is *"application is online*". [15,11.]

Furthermore, it is important to cache all the required resources and harmonize the status of the application along with its associated resources with the web server when there is a connection. After the availability of a network is checked and the resources are harmonized, the next step is to design the UI.

3.2 Designing the User Interface

When designing an offline application, there are many important aspects that need to be considered. Determining the number of hits or visitors to the website, page refreshing, coherency and eviction are among some. The importance of hit determination is that it enables one to assess the data to be cached and decide the efficiency of the website. [22.]

Moreover, since a data stored locally become out of date, refreshing the cached data is essential. Also, deciding the data to be changed from the locally stored data enables to get sufficient memory for fresh data. [22.]

Furthermore, determining the best way how local storage will unite data is useful since it is important to update data before sending them to queries. Figure 4 below depicts an extended layer of an offline web application's core elements.

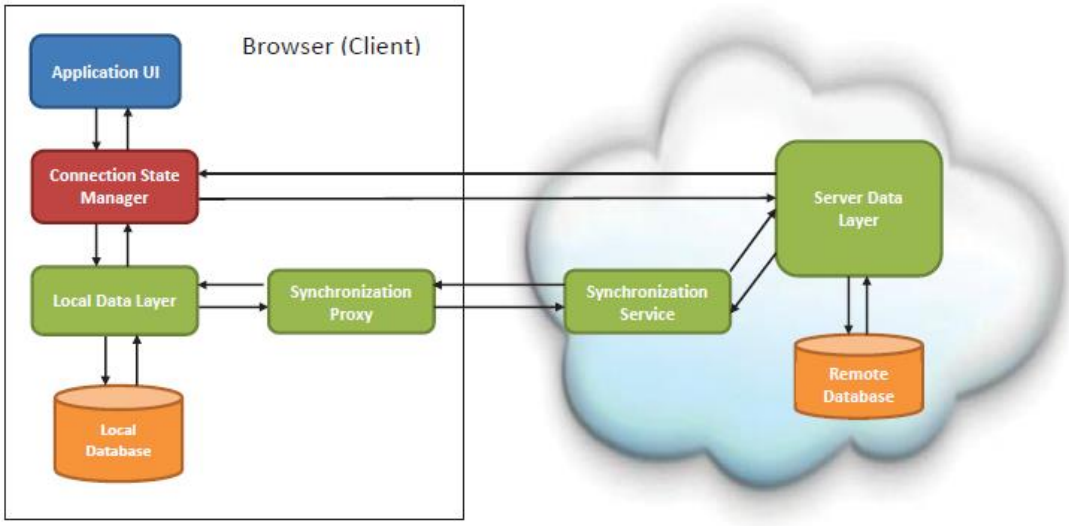

Figure 5. Advanced version of core offline web application elements. Copied from IBM DevelpersWorks [23]

As shown in Figure 5 above, designing an offline web application requires synchronizing the client and the remote applications. Hence, web developers need to give due attention when designing an offline application.

Even though performance is vital and needs to be considered in any application, it is a great concern for developers, especially when designing a web application that can be accessed by many people at the same time. When compared to the traditional web applications, desktop applications are faster and more responsive due to the fact that a large part of desktop applications exists in the client device and use local resources. Hence, to enhance a web application's performance, either a fast bandwidth connection is required or some part of the application data reside on the client machine, which in turn reduces the round trips to the server. [14.]

Some of the most important things that need to be considered when designing web applications for performance are as follows:

- ► Data caching
- ► Content Delivery Network (CDN)
- ► Optimizing Network Communication
- ► Use threads efficiently
- ► Using Transaction Effectively
- ► Optimizing Application Start Time
- ► Efficient Resource Management
- ► Hypertext Transfer Protocol (HTTP) Compression
- ► Content Expiration
- ► Pre-fetching resources

Hence, web developers need to consider the above lists when designing web applications in order to increase the performance and improve UX.[14.]

3.3 HTML5 Application Cache

To create a web application that provides good features requires approval from a web standard that are used frequently like HTML5, CSS3, JavaScript and other frameworks. Particularly, HTML5 has become a standard to responsive web applications offering a pleasant UI that works across most devices and major browsers.

The main importance of application cache is to save web files such as HTML, CSS, JavaScript and images on the user's machine. Storing these files in the user device is important especially during offline web development since they can be accessed even without a connection to the network. [15, 9.] Furthermore, application cache improves the performance of the application avoiding the round trips to the web server.

In order to create a simple offline application, it is important to declare a manifest file that contains a list of all resources that are required to be stored on the client device. Hence, when the user browses a web application for the first time, the application loads the manifest file along with the files listed on it into the client machine and saves them on the local device for subsequent requests. If there is no change in the manifest files, the downloaded resources will be maintained on the client device. If there is any modification in the files listed under the manifest or a change in manifest file itself, it will be downloaded again. [15, 11-15.]

Moreover, the manifest file needs Multipurpose Internet Mail Extensions (MIME) type of cache-manifest or text and requires a *.manifest* or *.appcache* extension. In addition, the first line of the manifest file should contain the text CACHE MANIFEST [15,12]. Listing 3 below shows how a manifest file is declared in HTML:

```
<! DOCTYPE HTML>
<html manifest="/cache.appcache">
<head>
...
</head>
<body>
...
</body>
</html>
```

Listing 3. Declaration of manifest file in HTML

As illustrated in Listing 3, inside the html head tag, the manifest file is declared and its extension is *.appcache*. Moreover, Listing 4 includes the contents of a sample manifest file.

```
CACHE MANIFEST
# 2012-02-21 v1.0.0

# Explicitly cached entries
CACHE:
index.html
css/style.css
images/logo.png
scripts/main.js

# Will be displayed if the user is offline.
FALLBACK:
/ /offline.html

# All other files require the user to be online.
NETWORK:
*
```

Listing 4. Example of a Manifest File

As shown in Listing 4, there are three different sections in the manifest file: CACHE, NETWORK, and FALLBACK [15,12]. Each of these sections will be explained in detail below.

CACHE

Cache is the default section of the manifest file and the browser downloads all resources that are included here when the user requests the site for the first time. In the above case, there are four files which are listed under the cache sub-heading and hence, the browser downloads all of them when loading the manifest file from the root directory of the website. Once they are downloaded, the resources will be available even if there is no connection to the internet. [15, 12.]

NETWORK

Resources listed under the sub-heading network require a connection to the internet. They depend on the network and get loaded from the web server when there is an internet connection. Most of the time, sensitive and secure files are listed under network section of the manifest file in order to protect them from caching in local devices instead users access these resources directly from the web server. Under the network section above, an asterisk (*) is used instead of listing the files, which means, files which are not mentioned under the cache section will be retrieved via an internet. Nonetheless, it is essential to possess a fallback action to enable the application to be operational in situations when a connection to the internet is unavailable. [15, 12-13.]

FALLBACK

The fallback section of the manifest file is important as a means of guaranteeing that resources that fail either due to a network connection, web server can be substituted by another locally available resource. For example, in the above case offline.html can be used as a replacement for these kinds of problems. [15, 13.]

Finally, even in situations where the application is online, those resources which are cached will always be used from the cache and not from the web server. Furthermore, cached files will be updated only when the manifest file is modified.[15, 15.]

3.4 Data Persistence

When building an offline application, there are two approaches to data persistence that need to be adopted: optimistic and pessimistic approaches. In the first approach, web applications let the modification of data realizing the application returns to online mode so that it will be harmonized with the server without creating any data conflicts. In contrast, the pessimistic approach assumes any type of data update most likely creates a conflict and hence it only reads data and does not allow any modification. [17.]

As a result, for offline web applications to work as required, it is advisable to follow the optimistic approach for data persistence. Nonetheless, in case if there is conflict when making changes to the data, it is recommended for the application to detect and manage the problem properly. Doing so enables one to share the data created by one user to another user simultaneously [17].

Furthermore, since the integrity and reliability of data are important, it is essential to perform data updates in a good way even in cases where immediate consistency is not required. Therefore, when designing an application that can be accessed offline, the system needs to synchronize automatically when it is online to reconcile between the web server and client. Moreover, if the synchronization process fails due to some reasons, it will resume without any data loss.

3.5 Database Synchronization

Using application that works offline means having permission to web information even in cases where there is no connection to the internet but users need to access their data. Moreover, all data updates are made to the local device of the client and only synchronized to the remote server when the network is available. However, though storing data locally solves the network problem, it has also some challenges [23]. Some of the problems and possible remedies are discussed in the following.

Consistency and responsiveness

Data consistency contradicts with application responsiveness in web development and hence web builders need to consider these two requirements and try to reconcile them. Data uniformity between the local database and web server is achieved by designing an application that stops different users from changing or requesting data. Nevertheless, it is hard to meet this condition since HTML is a stateless protocol. In addition, though storing some portion of server data in the local database helps to improve responsiveness, it creates cooperation of data integrity and consistency. [23.]

Client-side transaction logging

A record can be created by the client device when data is changed. When a network exists, this log can be sent to the web server for reconciliation [23].

Receiving server updates

There are two ways by which clients receive data from the server. When the application runs in the client browser for the first time, the web server sends all the required data. The second way is when the client already has data but the web server sends the same data again. In the second case, it is important to know the age of the information available in the device before making any updates to protect a data conflict whereas there is no problem in the first case since the data is new.[23.]

Change tracking

When the data available in the local device is modified, it is advisable to create a mechanism that tracks the change and reflects the information available in the server. Using this facility, any modifications like updates, deletions and inserts are traced and a failure to do so results in overwriting what has been done by other users. Hence, to solve problems arising from data modifications, change tracking is essential and one way is using record or row versions where every table column in the database is represented by the row's version number. [23.]

Conflict detection and resolution

Conflicts happen when two or more users try to change the same data and then attempt to apply the modifications to the database simultaneously. One way to resolve this problem is to allow only the latest change to be accepted by the web server [23].

Background synchronization

In order to allow the client use the user interface when performing data synchronization in an offline web application, it is recommended to design the synchronization to be done in the background [23].

Data exchange prioritization

While building a web application that works offline, it is essential to scrutinize all possible data exchange which means to enhance the transmission channel. Once the channel is identified and prioritized, data with high priority will be synchronized with a server when a connection is available and the one with less importance can be synchronize later on. [23.]

Security

Some precaution measures need to be considered when building an offline web application:

- Encrypting the database.
- Setting authentications to get access to the database.
- Synchronizing an encrypted data before sending to the database.

Generally, tackling the problems described above and implementing the remedies help to develop the best offline web applications. Hence, database synchronization is very important for offline website development and designers need to consider it seriously.[23.]

3.6 Privacy and Security

Currently, most of the day-to-day operations of individuals are online due to the expansion of an internet technology creating complications in the web industry. Nevertheless, most of these websites ask the user to enter personal information to gain access from the system though the information is sensitive. As a result, the consequence of this advancement in the internet technology has a negative effect on the privacy and security of the customer.[24.]

3.6.1 Privacy

Due to reasons mentioned above, application developers need to carefully build a website that is not vulnerable to hackers and keeps users' information secure. Some of the recommended ways to solve privacy issues during web design are cited below.

Privacy by design

Take privacy as a critical issue during design and do not wait to fix later when it occurs.

User centered-design

Since the primary beneficiaries of web applications are people, every design should be directed to users in order to give them full confidence in the security of the site. This can be achieved by designing an application that receives only necessary information from the user and by encrypting sensitive user data when a need arises. [25.]

Information confidentiality

One way to keep information confidential during data transmission over the internet is to use Hyper Text Transfer Protocol Secure (HTTPS). Furthermore, using a secure database to store the data enables one to protect the resources from attackers. [25.]

Control and log access

Controlling personal information by setting log access helps to track problems in cases when data misuse or loss happens.

To summarize, web developers need to consider customers' personal information with special care mostly when they are sensitive and misuse may result in privacy-related problems.

3.6.2 Security

In every part of the world, the role of web applications in performing day-to-day activities of various business organizations is paramount. It helps the economy by reducing cost and improving the speed of business operations. Nevertheless, this has also increased the risk to users' information and malicious attacks are being used every day, directed at web applications so as to affect the smooth operation of the business processes.

As a result, due considerations should be taken into account when designing a website particularly when identifying the entry points of security threats [25]. Hence, after the threats are examined and remedial measures are proposed, it is wise to design the web application since it mitigates the risk. The most known attacks to websites are cross-site scripting (XSS) and an SQL injection. Therefore, the application should be built to handle these two attacks by validating input when entered into the system.[25.]

SQL injection is the most common and dangerous attack to the website and advantage of the weakness created when validating input by sending unnecessary SQL queries to the database. Hence, using this advantage, hackers use an SQL injection to gain access to the database and take important information. [25.]

One way to protect information from an SQL injection is to always validate data, check its format and should be escaped before it is included in an SQL statement. Moreover, another way of safeguarding data from this attack is to apply a list privilege principle by which users with a low right to a database needs to operate their work in the database. [25.]

On the other hand, cross-site scripting uses a script injection error on a website by forwarding the information sends by the client to hackers. A cross-site attack can be non-persistent, which is the most common attack, or persistent. Hence, it is recommended to apply the same-origin policy when using client-side scripting languages to minimize problems arising from cross-site scripts. [25.]

To conclude, in this part of the study, different materials used to build offline web applications are discussed. Requirements to develop offline applications, design issues, HTML5 application cache and manifest files were some of the topics explained in detail. In addition, data persistence, database synchronization and privacy and security related issues were raised and possible remedies to solve associated problems were explained.

4 Result

To achieve the planned goal of the project, both the frontend and the backend were fully developed based on the requirement set in chapter 3. HTML5, CSS and JavaScript were used for the client-side and PHP and MySQL database were applied for the server-side. Moreover, an admin page was built to manage the contents posted on the front page and for security purposes.

The client-side contains a header, a menu bar, a content area, a sidebar and footer. The menu bar contains links and a search box, the content area contains images and texts and the sidebar contains a list of recent posts with a title and images. All these contents are populated directly by the administrator. A screenshot of the admin login page is shown in Figure 6 below:

Figure 6. Screenshot of admin login page

As illustrated in Figure 6, the website administrator has a user name and password to login into the page before performing any operations. After successfully logging into the page, the administrator sees the main page as shown in Figure 7 below:

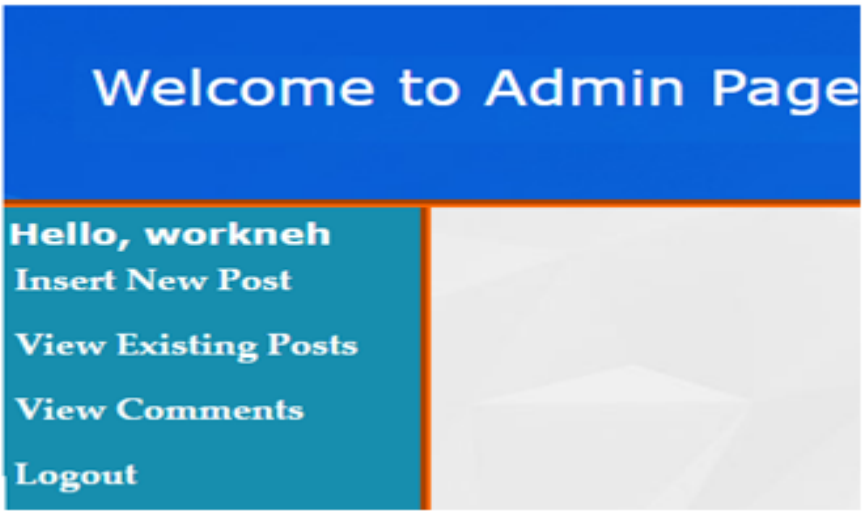

Figure 7. Screenshot of admin index page

As witnessed in Figure 7, the administrator can perform different tasks on this page. Inserting a new post into the website, viewing existing posts and comments are some of the operations. For example, the results of opening the insert new post page is shown in Figure 8 below:

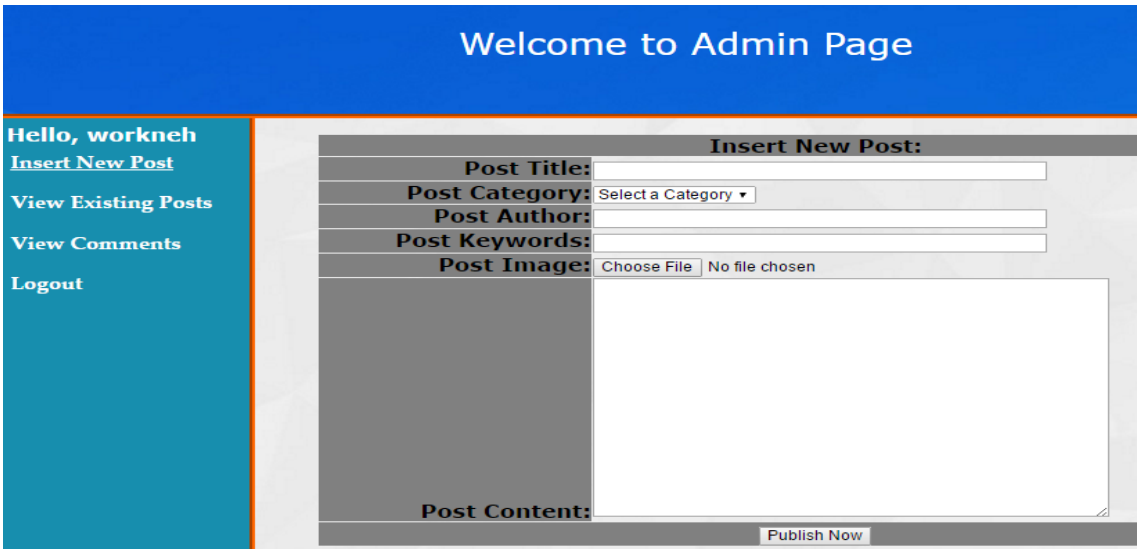

Figure 8. Screenshot of insert post page

As illustrated in Figure 8, the insert page has a post title, post category, author, keywords for search purposes, post image and content fields. After inputting all these fields, the post will be published directly to the website and to the database. If the administrator tries to publish the page without inputting the fields, a JavaScript alert message will be fired. Figure 9 belowdepictsa viewexistingpostspage.

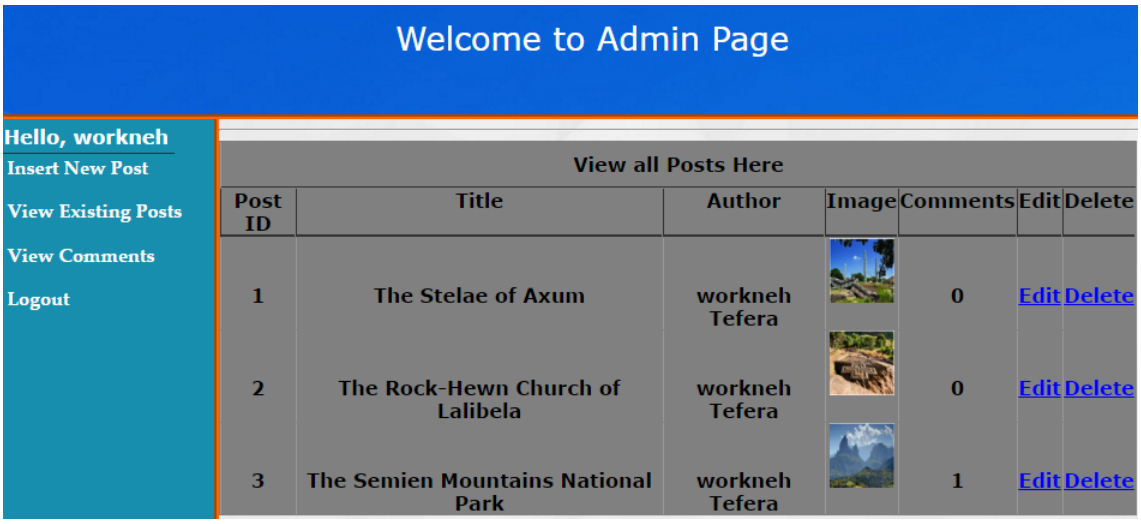

Figure 9. Screenshot of view existing posts page

As shown in Figure 9, there are three posts with their respective post id, title, author, image and comments. Moreover, there are two additional columns for edit and delete. If the data posted into the website is not proper, the administrator can either edit or delete the content in this page. There is another view comments page where the administrator can manage a user's comments before posting it into the website as shown in Figure 10 below:

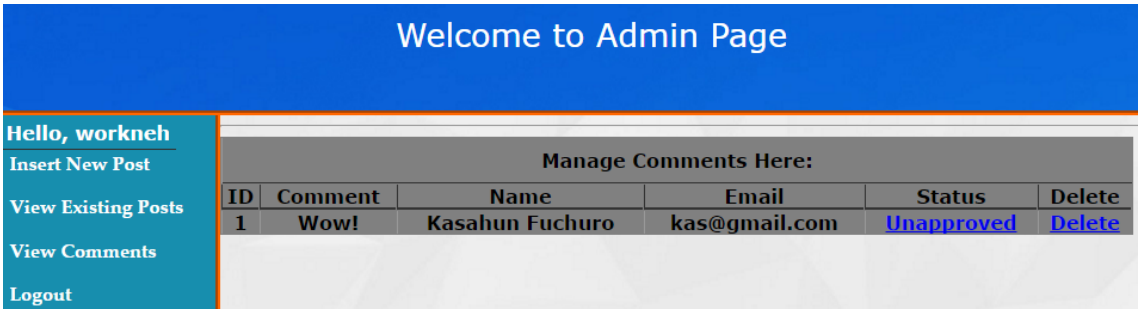

Figure 10. Screenshot of view comments page

As shown in Figure 10, the view comments page has id, comment, name and email of the person who made the comment, status and delete action. The administrator can either approve or delete the comment in this page. Approved comments can be seen on the website.

After the administrator successfully accomplishes all the above tasks, the update will be shown on the client-side. A screenshot of the front page of the website is shown in Figure 11:

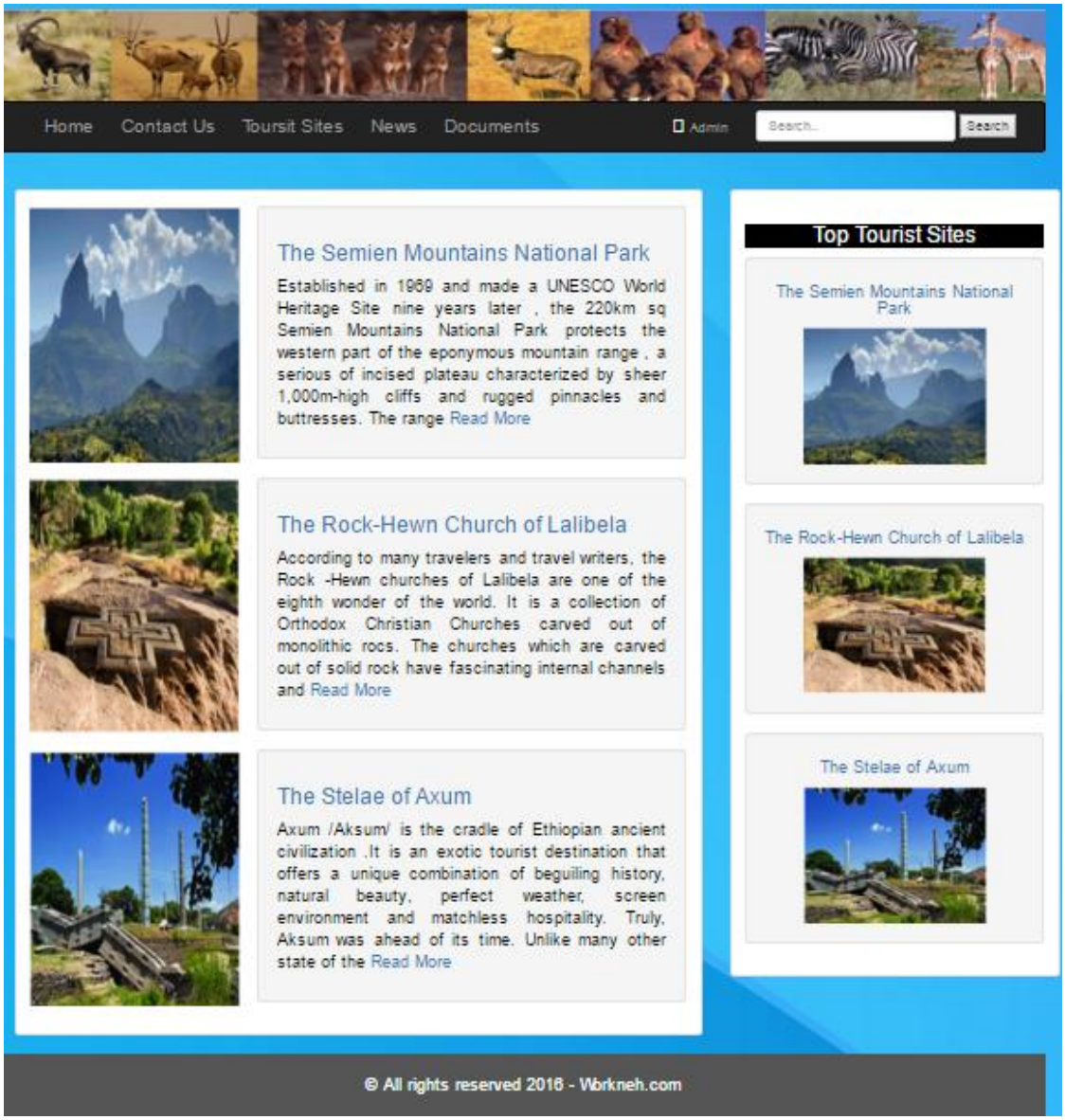

Figure 11. Front page of the website

As depicted in Figure 11, the front page includes a logo, menu, search box, main content post area, sidebar and footer. Each post contains one image and texts while the sidebar contains a small-size image and title of the posts. If the user clicks on the read more link or on the title of the post or sidebar, a detail page will be opened as demonstrated in Figure 12 below.

Figure 12. Details page of the website

As shown in Figure 12, a detail of the post's text and a larger size image are displayed on a separate page. In addition, comments made by the user are posted along with an empty form.

After all the pages were built as per the requirement, the website was tested. Viewing the front page, detail page and searching for an item within the page on the client-side are possible. Moreover, clients can give feedback on the web page after viewing the detail page. However, the comments can be posted after the administrator approves them. All these functionalities were checked with an internet connection.

After the online version of the website had been tested as required, offline functionality was included to the web page and tested it again without an internet connection. All files that were considered to be offline were listed in the manifest file and the name of the manifest file was included in a HTML tag of those pages. Then all the documents including the manifest file were put on an online web server (mysql.metropolia.fi) to test its functionality without a network connection.

The final result of the test was that the website was working perfectly as planned without any problem. All files listed in the manifest file were cached and Figure 13 below shows a list of files cached in a chrome browser:

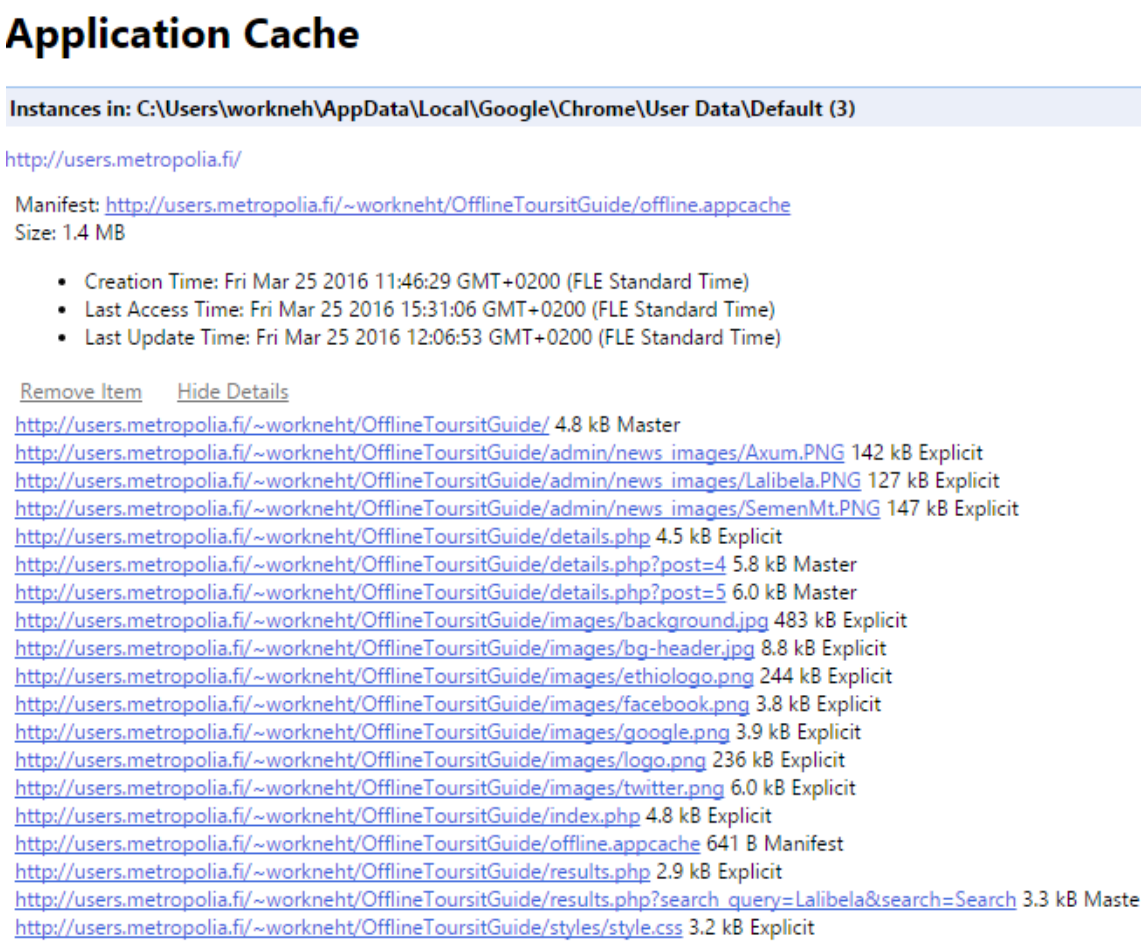

Figure 13. List of files cached in chrome browser

To conclude, Figure 13 above illustrates that browsers cache files listed in the application cache due to the HTML5 capability to give such services to store data offline. Hence, the goal of the study is achieved and the application is working as intended.

5 Discussion

5.1 Problems Encountered

Although most web application development methods explained in this project are possible, designing and building an offline website has many challenges. One of these problems is the lack of an easy way to manage the information captured when the client is offline. This is because users that are online may modify temporary data on their devices without letting other users know even if the data is potentially in use in some offline applications. It could be good if there were ways to identify which data is needed by the database before the user deletes them from the browser. Furthermore, sometimes users may not know whether they are actually running offline or not unless there is a mechanism that notifies them.

Moreover, sometimes users do not know where cached data exists in their machine due to variation in the implementation of many web browsers and platforms. Due to the difference in browsers in storing cached files in different locations, users get confused when opening different browsers to access offline applications.

Furthermore, to allow synchronization and work successfully in offline web applications, the application has to be opened since there is no regular synchronization, especially in the Windows service. This is because all JavaScript code execution takes place after opening the browser.

In addition, due to limits in the storage space of the client disk, there is a difficulty in storing a large amount of cached data and the user runs out of space after saving some data.

5.2 Solutions Sought

The possible solutions that enable to reduce the above and other related problems during offline web applications development are discussed in detail in the following part.

a) Web design considerations

During web design, it is essential to know the users and how they use web applications. Moreover, having knowledge about users' actions on the application and the rate of occurrence are important considerations. This information is important to identify the pages that need maximization when used offline. One way to understand this is simply observing a server log.

Another consideration when designing a website is to check how long it will take to load each page when the request is fired. It is recommended that every page should load faster to optimize user experience. Developing a web application that exists offline does not imply slow page loading. Particularly when the application is online it will be busy since it takes time to arrange itself for offline use. In addition, it is important to make sure that any pre-fetching or background loading should be performed only after the page is loaded and have limited or no effect on web application performance.

Furthermore, storing the right and the required data is another design consideration. Storing the right data means some unsafe resources should not be saved on the browser mainly when they can be retrieved from a public domain. This is particularly true when storing time-sensitive resources on the client browser that take a longer time for updates to obtain the right version of the information

The last design consideration is to be a good web user and avoid running many applications at the same time. This is because mostly many web users run a large number of applications in parallel, which negatively affects the bandwidth of the web application. Hence, it is recommended to store only important information that is more possibly to be used when going offline.

b) Implementation Risks

Problems related to implementation seriously impact offline web services. Some of the issues that need due attention by web developers include the following.

Security and privacy

As stated in the methods and materials 3.6, the sensitivity and privacy of an information stored in the client browser have to be given greater priority and kept safe from malicious attackers.

Regulatory compliance

Ensuring the fulfillment of conditions set by web standard that the data stored in the client device should be accessed publicly only when it meets the requirements. There are some industries like banks, health services and some government offices set strict rules to their customers when offering their services online.

c) Browser Differences

Currently, it is the W3C that sets standards on different web browser vendors to guarantee the development of the web. It is the governing organ that ensures browser suppliers fulfill some standards to build a compatible environment. Nonetheless, it is often difficult to adhere to the rules as some vendors produce incompatible browsers which create a problem for web developers when building a web application that works across different browsers. Moreover, it increases application development time and cost by reducing a developer's efficiency.

d) Performance and Scalability

When building a web application, developers need to consider performance and scalability seriously as long as the application is used by a number of users at the same time. Achieving high performance is vital which in turn results in increased scalability based on the application process environment. Hence, to increase application performance, it is important to run the web application on devices that have more resources. For instance, application performance doubles if the application runs

on a computer that doubles its resources. Practically, however, this does not always work. Rather the applications work better if they run in environments that have more resources when the workload is bigger.

Application scalability can be measured in many ways. In order to achieve higher application scalability, security and client-side development and debugging should be considered. The effect of security on application scalability and performance can be easily witnessed when setting input validation rules to a web page form. Sending a page with a set of security rules in the form of input validation relatively decreases the performance of the page when compared to a page without validation. However, whatsoever is its impact on the performance of the web pages, priority should be given to application security so as to tackle malicious attacks.

Finally, client-side debugging tools are preinstalled in most web browsers or can be added to these browsers as extensions. The advantage of these tools is to examine web storage or indexedDB contents in addition to its capability to compile HTML, CSS, and JavaScript together.

6 Conclusions

To develop web applications using HTML5, it is essential to observe the client and server side components. Different web techniques such as good coding practice, client and web server caching and resource compression enable to build responsive and high-performance web applications that increase user experience.

Most of the time web applications require a network connection to operate smoothly. Nevertheless, the availability of an internet connection is not usually assured though the intention of web users is always to use the applications regardless of the problem. As a result, developing an application that solves this problem and always works without being dependent on a network connection is important.

In this thesis, different ways of designing web applications were discussed with due emphasis on using the new features of HTML5 to develop an offline web application in order to increase user experience. Moreover, web application tools such as HTML5, local storage, indexedDB and other APIs were presented and explained in detail. In addition, techniques like data synchronization, security, performance and scalability and others that improve user experience being looked at with greater explanation.

To conclude, in this project, a website that works even when the user is not connected to the network was developed using application cache and local storage and hence, the intended aim of the project was achieved. Nonetheless, the technologies are not universally supported, require an understanding of their vagaries and are still to some extent under development. Therefore, it is recommended that web developers to use this technology in order to increase user experience (UX) and future students to conduct broadened research on HTML5’s offline functionality.

References

1. K. L. James. The internet: A User's Guide. Prentice-Hall, August 1, 2004.

2. Gary B.Shelly, Mark, F. Web 2.0: Concepts and Applications. Cengage Learning, March 3, 2010.

3. Core elements of online web applications[online]. Webreference; Created: March 27, 2003, Revised: October 3, 2005
   URL: http://www.webreference.com/programming/php_mysql2/index.Accessed 20 February 2016.

4. Marco, C., Peter, E., Charles, B., Nathalie, W., Cyril, H. HTML5 Solutions: Essential techniques for HTML5 Developers. Apress, October 19, 2011.

5. Hunter Medney. Core elements of offline web applications [online]. IBM DevelopersWork; 08 May 2007
   URL: http://www.ibm.com/developerworks/lotus/library/expeditor-offline/.
   Accessed 21 February 2016.

6. Flickenger, R. How to Accelerate Your Internet: A Practical Guide to Bandwidth Management and Optimization Using Open Source Software. INASP, 2006.

7. Jeremy McPeak, Paul Wilton. Beginning JavaScript (fifth edition). Wrox, March 17, 2015.

8. Rob Crowther. Hello! HTML5 & CSS3: A user-friendly reference guide. Manning Publications, October 29, 2012.

9. Adam Freeman. The Definitive Guide to HTML5. Apress, December 14, 2011.

10. The HTML DOM Tree [online]. W3Schools.com; Created 1999.
    URL: http://www.w3schools.com/js/js_dom.asp.
    Accessed 22 February 2016.

11. S. V. Nagaraj. Web Caching and its Application. KluwerAcademicPublishers, May 1, 2004.

12. Neil Edde. Website Optimization: An Hour a Day. Sybex, an Imprint of Wiley January 1, 2012.

13. Eric Freeman, Elisabeth Robson. Head First HTML5 programming: Building Web Apps with JavaScript. O'Reilly Media, October 1, 2011.

14. Shiju Varghese. Development with Go: Building Scalable Web Apps and RESTful Services. Apress, September 15, 2001.

15. Nilachala P. Developing Offline Web Applications using HTML. URL: http://www.tcs.com/SiteCollectionDocuments/WhitePapers/TEG_Whitepaper_Developing_Offline_Web_Application_Using_HTML5_0212-1.pdf. Accessed 22 February 2016.

16. Mike Shema. Hacking Web Apps: Detecting and Preventing Web Application Security Problems. Syngress, September 12, 2012.

17. Adam McDaniel. HTML5: Your visual blueprint for designing rich web pages and applications. Visual, October 13, 2011.

18. Mark Lassoff. Mobile App Development with HTML5. LearnToProgram, Incorporated, Mach 10, 2015.

19. Damon Oehlman. Pro Android Web Apps: Develop for Android Using HTML5, CSS and JavaScript. Apress, February 21, 2011.

20. Len Bass, Paul Clements, Rick Kazman. Software Architecture in practice (third edition). PearsonEducation, September 25, 2012.

21. High level HTML5 Architecture [online].HTML5 Web Application. Micronet Techno, 10 April 2001. URL: http://www.micronet-techno.co.jp/en/sol_html5_e.html. Accessed 23 February 2016.

22. Kroeger, R. Cache Pattern for Offline Web Applications. Retrieved from https://dl.google.com/io/pres/W_0145_CachePatternforOfflineWebApplications.pdf. February 25, 2016.

23. Hunter Medney. Core elements of offline web applications [online]. IBM DevelopersWork; 08 May 2007. URL: http://www.ibm.com/developerworks/lotus/library/expeditor-offline/. Accessed 24 February 2016.

24. Nigel Waite, Christine Ennew. Financial Services Marketing. Routledge, November 13, 2006.

25. Microsoft Corporation. Improving Web Application Security: Threats and Countermeasures. Microsoft Press, January 31, 2011.

Appendix 1. Index Page Used for the Website Presentation

Index.php

```
<!DOCTYPE html>
<html>
<html manifest ="offline.appcache">
    <head>
        <title> Tourist Guide Website</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="styles/bootstrap.min.css">
        <script src="js/jquery.min.js"></script>
        <script src="js/bootstrap.min.js"></script>
    </head>
    <body>
    <body>
        <div class="container text-center">
           <div class="row">
                <div class="col-lg-12">
                     <a href="index.php" ><img src="images/logo.png" width="1140px" height="180" id = "logo" /></a>
                </div>
            </div>
        </div>
        <div class="container text-center">
            <div class="row">
                <div class="col-lg-12" >
                   <nav class="navbar navbar-inverse">
                        <div class="container-fluid">
                            <div class="navbar-header">
                                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">
                                    <span class="icon-bar"></span>
                                    <span class="icon-bar"></span>
                                    <span class="icon-bar"></span>
                                </button>
                            </div>
                            <div class="collapse navbar-collapse" id="myNavbar">
                                <ul class="nav navbar-nav" id="catnavbar">
                                    <?php  include("includes/navbar.php");  ?>
                                </ul>
                                <form class="navbar-form navbar-right" method ="get" action="results.php" >
                                    <input type="text" name="search_query" class="form-control" placeholder="Search..">
                                        <span> <input type="submit" name="search" value="Search" /></span>
                                </form>
                                <ul class="nav navbar-nav navbar-right">
                                    <li><a href="admin/login.php"><span  class="glyphicon glyphicon-user"></span></a></li>
                                </ul>
                            </div>
                        </div>
                    </nav>
                </div>
            </div>
        </div>
        <div class="container">
            <div class="row" style="margin-top:20px">
                <div class="col-lg-8" >
                    <div class="panel panel-default text-left">
                        <div class="panel-body">
                            <?php  include("includes/post_content.php");  ?>
                        </div>
                    </div>
                </div>
                <div class="col-lg-4">
                    <div class="panel panel-default text-center">
                        <div class="panel-body">
                            <?php  include("includes/sidebar.php");  ?>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <footer class="container-fluid text-center">
          <h4>&copy; All rights reserved 2016 - Workneh.com</h4>
        </footer>
    </body>
</html>
```

Appendix 2. Content Page Used for the Website Presentation

Content_page.php

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title> Tourist Guide Website</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="styles/bootstrap.min.css">
        <script src="js/jquery.min.js"></script>
        <script src="js/bootstrap.min.js"></script>
    </head>
    <body>
        <?php
            if(!isset($_GET['cat'])){
                $get_posts = "select * from posts order by rand() LIMIT 0,5";
                $run_posts= mysqli_query($conn, $get_posts);

                while($row_posts = mysqli_fetch_array($run_posts)){
                    $post_id= $row_posts['post_id'];
                    $post_title= $row_posts['post_title'];
                    $post_date= $row_posts['post_date'];
                    $post_author= $row_posts['post_author'];
                    $post_image= $row_posts['post_image'];
                    $post_content= substr($row_posts['post_content'],0,300);

                        echo "
                        <div class='row'>
                        <div class='col-sm-4'>
                                <img  src ='admin/news_images/$post_image' width='230' height='288' />
                        </div>
                        <div class='col-sm-8'>
                          <div class='well'>
                        <h3 ><a href='details.php?post=$post_id'>$post_title</a></h3>
                            <p >$post_content<a href='details.php?post=$post_id'>Read More</a></p>
                        </div>
                         </div>
                        </div>    ";
                    }
                }
            else{
                if(isset ($_GET['cat'])){
                    $cat_id=$_GET['cat'];
                    $get_posts = "select * from posts where category_id='$cat_id'";
                    $run_posts= mysqli_query($conn, $get_posts);

                    while($row_posts = mysqli_fetch_array($run_posts)){
                        $post_id= $row_posts['post_id'];
                        $post_title= $row_posts['post_title'];
                        $post_date= $row_posts['post_date'];
                        $post_author= $row_posts['post_author'];
                        $post_image= $row_posts['post_image'];
                        $post_content= substr($row_posts['post_content'],0,100);

                    echo "<div class='row'>
                    <div class='col-lg-12'>
                        <div class='panel panel-default text-left'>
                            <div class='panel-body'>
                                <h3 ><a href='details.php?post=$post_id'>$post_title</a></h3>
                                <img  src ='admin/news_images/$post_image' width='100' height='100' />
                                <span style='font-size:18px'>$post_content<a href='details.php?post=$post_id'></a></span>
                        </div>
                        </div>
                    </div>
                </div>  ";
                    }
                }
            }
        ?>
</body>
</html>
```

Appendix 3. Sidebar Page Used for the Website Presentation

Sidebar.php

```php
<div class="sidebar">
    <div style='background-color:black'><h3 style='color:white'>Top Tourist Sites</h3></div>


        <?php
            $get_posts = "select * from posts order by 1 DESC LIMIT 0,5";
            $run_posts= mysqli_query($conn, $get_posts);

            while($row_posts = mysqli_fetch_array($run_posts)){
                $post_id= $row_posts['post_id'];
                $post_title= $row_posts['post_title'];
                $post_image= $row_posts['post_image'];

                echo "

                    <div class='well'>
                        <h4><a  href='details.php?post=$post_id'>$post_title</a></h4>
                        <img  src ='admin/news_images/$post_image' width='200' height='150' />
                    </div>
                ";

            }
        ?>
</div>
```

Appendix 4. Application Cache File Used for Offline Functionality

Offline.appcache

```
CACHE MANIFEST
#This is final version is now working perfect

CACHE:
index.php
details.php
results.php
styles/bootstrap.min.css
js/jquery.min.js
js/bootstrap.min.js
images/background.jpg
images/ethiologo.png
images/logo.png
admin/news_images/Axum.PNG
admin/news_images/Lalibela.PNG
admin/news_images/SemenMt.PNG
```