Course Title: Capstone Design
Submission Data: December 2017
Prepared by: OUMNYA EL Jamaly
Supervised by: Dr. TAJJEEDINE RACHIDI
HISTORY AROUND ME - AN ANDROID MOBILE APPLICATION

Capstone Report

Student Statement:

I, the designer of this project, affirm having applied ethics to the design process and in the selection of the final proposed design. And that I have held the safety of the public to be paramount and have addressed this in the presented design wherever may be applicable.

________________________OUMNYA EL JAMALY__________________

Approved by the Supervisor

________________________Dr. TAJJEEDINE RACHIDI__________________
Acknowledgement

The personal efforts invested in this project have but a mere quota of credit for its successful completion; in fact, I believe the credit should go to the love, support, encouragement, guidance and indulgence of the many individuals in my life.

I therefore would like to take this opportunity to express my appreciation to these people without whom this project would not have seen the light nor a successful closure.

I would like to first address my deepest gratitude to Dr. Tajjedine Rachidi, first for agreeing to be my capstone supervisor, for his assistance and counsel throughout the project lifespan from the selection of the project topic to the conclusion of the project, for his leniency toward my oftentimes lack of communication and progress, and finally for his faith in my ability to deliver a quality project, which in turns gave me the confidence to carry on with the project and its conclusion.

I would also like to express my most unbound and profound gratitude to my loving parents and dear friends for their unconditional love and support, their faith in me even when I did not have any, their precious and wise counsel, and the wonderful moments together. Moments I hold dear to my heart and will forever carry with me. indeed, I would have never made it this far without them.

And last but not least I would like to thank God for blessing me with a loving family, wonderful friends, and most interesting, engaging and life-altering acquaintances, for providing me with life-defining opportunities, one of them being able to study in a prestigious university such as AUI, lastly and most importantly for peace and harmony in a time of conflict, war and despair.
List of Figures

Figure 2.2.2.1 - Screenshots of History Here US Mobile App
Figure 3.1 - Illustration of the Software Engineering Life Cycle
Figure 3.2.1 – Illustration of the Prototyping Model
Figure 4.1.1.1 The Use Case Diagram
Figure 4.3.1.1 The Class Diagram
Figure 5.1.1.1 Android Studio Logo
Figure 5.1.1.2 Java Logo
Figure 5.1.1.3 XML Logo
Figure 5.1.2.1 Firebase Logo
Figure 5.1.2.2 Firebase Crash Reporting Screenshot Example
Figure 5.2.1 Attempted System Architecture
Figure 5.2.2 Enhanced System Architecture
Figure 5.4.1 The Activity Diagram
Figure 5.5.1 The Sequence Diagram
Figure 6.1.1 Firebase Android Studio Assistant Tool Screenshot
Figure 6.1.2 Firebase Authentication Console Web Interface

List of Tables

Table 3.3.1 Time Management Planning Table
Table 4.3.1 Historical Site attributes and description
Table 4.3.2 User attributes and description

List of Acronyms and Abbreviations

IDE - Integrated Development Environment)
Contents
Acknowledgement ........................................................................................................ iii
List of Figures ................................................................................................................. iv
List of Tables .................................................................................................................. iv
Abstract ......................................................................................................................... vi
1. Introduction .................................................................................................................... 1
2. Project Scope ................................................................................................................ 2
   2.1. Project Overview .................................................................................................... 2
   2.2. Market Information ............................................................................................... 2
3. Project Methodology .................................................................................................... 4
   3.1. Feasibility Study ................................................................................................... 5
   3.2. SE Process Model ................................................................................................. 8
   3.3. Project Management Plan ..................................................................................... 9
4. Project Requirement Specification .............................................................................. 9
   4.1. Functional Requirement ....................................................................................... 9
      4.1.1. Use Case Diagram ......................................................................................... 11
   4.2. Non-Functional Requirement ............................................................................... 11
      4.3. Database Analysis: ............................................................................................ 12
      4.3.1. Class Diagram ............................................................................................... 15
5. Project Design ............................................................................................................. 16
   5.1. Technology enablers ............................................................................................ 16
   5.2. System Architecture ............................................................................................ 22
   5.3. Database modeling .............................................................................................. 23
   5.4. Activity Diagram ................................................................................................ 25
   5.5. Sequence Diagram ............................................................................................... 27
6. Project Implementation ............................................................................................... 29
   6.1. Implementation process: .................................................................................... 29
7. Project Testing and Validation ................................................................................... 35
8. Steeple Analysis .......................................................................................................... 36
9. Conclusion ................................................................................................................... 37
10. Future work ................................................................................................................. 38
11. References ................................................................................................................ 39
Abstract

Solutions such as google maps and google places have provided the user with easier access to close by institutions and places of diverse use and interest and are even able to provide the user with relevant information on the desired locations; however, these solutions come short when it comes to providing the user with ACCURATE historical sites and their related background complete information. My emphasis on the word “accurate” is due to the fact that the other solutions out there such as Google Places, when prompted/queried for “Historical Sites,” give results that are far from being restricted to historical sites but go beyond to include non-historical locations such as regular restaurants or shops. In addition, locations where famous historical battles took place, or archeological sites, such as Jbel Irhoud where the oldest human remains were discovered, and their related information, are not provided by such solutions.

This project aims at producing a mobile application that would provide accurate information to the user, based on his/her geolocation.

The app will allow the user to visualize the surrounding historical sites, if any, and provide the related historical data.

After a meticulous Software Engineering process, the app will be developed for the android operating system using technologies such as Android studio and Firebase.
1. Introduction

Our Moroccan historical heritage is extensive and predates the twelve centuries existence of the state. For millennia, the land of our beloved country has had many ancient cultures and civilizations leave their fingerprints throughout the different stages of history. From Prehistory, Classical Antiquity, and Early Islamic areas, to the more recent fight for independence, the Moroccan historical and cultural heritage is rich to the extent that it has been the subject of study and interest of many globally acknowledged historians and archeologists. An interest that paid off quite remarkably only recently, while the scientific world believed the human race to be 200,000 old, a discovery in Morocco came to put an end to this theory when the oldest human (*Homo Sapiens*) remains ever dating to 300,000 years back were discovered in the archeological site of Jbel Irhoud, south east of Safi, and this is only the most recent and significant discovery [5]. Furthermore, UNESCO has recognized nine historical sites as World Heritage (*Patrimoine Mondial*) and Morocco has filed applications for thirteen more sites to be considered for the distinction. In addition, more than three hundred sites are classified as National Heritage (*Patrimoine National*) [6].

This historical legacy is not restricted to archeological sites or monuments but includes famous battles, fought across the country for religion, freedom and independence; although we can not see their remains, as it is the case for other types of sites, these significant pieces of history have shaped our country’s history and present.

Unfortunately, as Moroccan citizens we are very poorly informed and educated about our historical heritage.

Furthermore, despite the lack of sources for historical information, Morocco still manages to attract a high number of tourists from every corner of the world. Morocco has gained a reputation for being a land of generosity, a land of beauty and diversity and a melting point of cultures, religions and languages. Those aspects are experienced by tourists through exploring; however, the richness of our historical heritage will not be exploited to its fullest potential, unless the source of such information is made available and easily accessible.

For these reasons and more, the idea of a mobile application that would make this kind of information available and easily accessible becomes highly attractive and represents the right project idea for my capstone project.
1. Project Scope

1.1. Project Overview

This capstone project is in the form of a mobile application that informs the users of their neighboring historical sites and landmarks by first determining their Geo-position. The app also allows the user to have access to more information about the specific landmarks surrounding him if they choose to click on any of the sites icons on the map or list.

- **Suggested Project Title/App Name:** HISTORY AROUND ME.
- **Project Time Frame:** Since this project is to be completed as required by the Capstone Design Course, its time frame is this current FALL 2017 semester and it is to be delivered by the end of November and presented during the first week of December.

1.2. Market Information

1.2.1. Target Audience

The Targeted audience will be the Moroccan user: students, researchers, history lovers … etc. wanting to know more about their county’s historical heritage and not willing to miss out on the opportunity to discover those sites in person if they ever come across or are nearby any. In addition to the regular Moroccan traveler, the targeted audience also includes tourists traveling to Morocco with the purpose of discovering the culture, landscape and history of the country.

1.2.2. Competitors

During my research, I did not come across any similar software or app for the Moroccan context; however, there exists other cases of apps that offer not quite exact but similar services. They do so for their respective countries and do not extend their services to other countries.

An example of such apps is **History Here**. This application is geolocation based and is an interactive guide to thousands of historical locations across the US [13].
Other examples of such applications are some mobile applications that use **Google Places API**. This API belongs to the family of Google Maps APIs and allows building location aware apps. Which means that the apps using this API can provide users with location and information on places of interest on the map. For instance, we can build an app, embed the Google Places API and configure it to only display coffee places within a given perimeter, and when clicking on a certain coffee shop pin on the map, we are provided with a detailed information page listing the name of the place, pictures, description, contact information, and rating information.
2. Project Methodology

Figure 3.1

History Here Mobile Application is a software product, and before delving into the implementation we had to go through a meticulous Software engineering process just like we were taught in the software engineering classes. In order to start developing our application, it is necessary to have a clear picture on the feasibility of the project, its functionalities, requirements to be delivered and constraints to be taken into consideration, investigating the best suited technology for the task, and much more.

This process involves multiple stages, steps and modules in accordance with the divide and conquer approach to guarantee building a successful end product that satisfies user requirements and expectations and allows unchallenging extensibility and maintainability. The software engineering process is a cyclic one involving the following stages: Project Initiation and Analysis, Design, Implementation, Testing and Maintenance. First we needed to start by constructing a high level understanding of the project. Thus, after a brief introduction of the project topic and the motivation behind it, the very first step was to initiate the Analysis phase by conducting a Feasibility Study. The feasibility study allowed us to assess the achievability of the project from various aspects, namely Economical, Technical, Legal…etc. The next step was the requirement gathering which consisted in determining the functions that the application will need
to implement, the data it will need to work with and the constraints under which it would need to operate. These will translate into a detailed list of Functional and Non-Functional requirements in addition to the Data Analysis.

The second phase is the Design; this phase is very important as it represents the blueprints of our project. This stage uses the results of the Analysis phase and establishes an architecture and structure that will rule the implementation phase.

The third phase is the Implementation stage; this is where we actually delve into the making of our end product. This is where we use the chosen tools discussed in technology enablers to implement the functionalities set during the requirement gathering in accordance with the scheme and constraints set during the Design Phase.

The Testing is performed at first in parallel along with the implementation, testing the functionalities one or few at a time until the app is ready for a stable release. This is when a comprehensive testing is performed.

After the application is made available for use, the Maintenance phase comes as a follow up to fix whatever problem that might have occurred or expand the functionalities, performance and environment.

2.1. Feasibility Study

Within the framework of the software engineering analysis phase, a feasibility study is required to determine whether this project is feasible or not in terms of several factors including the economical, technical, and technological, in addition to the development time and relevant data and pertinent information to the context of our software.

As mentioned before, this software will be in the form of a mobile application. As such, its feasibility is greater due to the ever growing popularity and importance of mobile development in recent years. The importance lies in the fast and vast spreading of mobile technology usage. Nowadays, almost everyone on the planet carries a smart mobile phone that regulates every aspect of their lives through several and diverse applications installed on their device. We can say with certainty that no two phones on the planet hold the same set of applications, since the individual needs are simply different from one person to the next. This is why the need for an immense pool of applications to choose from has raised the demand for more and more apps of
diverse purposes to be developed. This need and popularity have driven companies like google and Apple to simplify the task for developers and individuals interested in mobile development by providing tools and environments free of charge, open source and easy to use. In addition, thanks to the assistance of the worldwide development community, learning new skills and technologies is ever more reachable due to the countless tutorial and learning platforms.

2.1.1. Technical Feasibility

Concerning the technical side, the ease and rate with which mobile applications are developed and deployed nowadays is substantial, and more efforts and services emerge every day to add more ease and flexibility to the process and make the rate of deployment even higher. In addition, many similar applications have been/are being published on Google Play, and they are successfully attracting the interest of the community.

Technological Feasibility

The technological side of the analysis is concerned with the availability, accessibility, and appropriate choice of the technologies to be used for the development process.

Before discussing the technological dilemma, a choice had to be made: whether to build a Hybrid or Native application. A hybrid app is one that is compatible across platforms and therefore has higher chances to reach a higher number of users, whereas, a native app is more specific and wired to work on one operating system. However, a choice was made to go for the Native option. This decision was made because we prioritized user experience, which happens to be the key factor for the application success and performance over the spread of use. Native apps are known to be quite performant in comparison to their hybrid competitors, and provide easier access to the build-in functionalities of the device. In addition, since the apps are built using the specific platform technical and user guidelines, their look and feel is consistent with other apps already on the device and to which the user has got used to.

A choice was also made to build the app for the Android operating system. This choice was encouraged by the fact that more than 80% smartphones are based on the android operating system including manufacturers such as Samsung, LG, HTC…etc. [14]. This app will be built for the Android OS using the Java programming language on the Android studio IDE for the front end. Concerning the back end, after evaluating several options, we opted for the Firebase Google
Web service as a full package back end solution for hosting, real time database and authentication in addition to many other interesting features.

Thus, in terms of the technical and technological analysis, this project is feasible, since the tools and platforms are available along with to the required resources to learn the skills needed for the development.

2.1.2. Economic feasibility

In terms of the economic cost of the development process and delivery, the cost is minimized. The analysis, design, implementation and maintenance of the app will be completely free. The only step that would require a cost to be paid is after the deployment process where a fee of $25 is paid to publish the app on Google Play. On the user side of the equation, the app will be free to download; and the user will only be concerned with the cost to access the internet - - as the app will be dynamic- and to access the server and Database, one will need to have internet connection. Thus, in terms of the economic analysis this project is feasible.

2.1.3. Legal Feasibility

As far as the legal aspect of this study is concerned, no legal relevance or obstacle was encountered. Thus, this project is feasible in terms of the Legal Analysis.

2.1.4. Scheduling Feasibility

The scheduling analysis is an important one to consider as it determines, given the availability of the conditions and tools, whether the development side will be successful in delivering the end product by the due date. I believe that given the scope of the project and the available time and resources, the delivery of an end product by the due date is feasible.

2.1.5. Data Availability

In terms of securing the right accurate data for the app, an interview with one of the university History professors, Dr. Driss Maghraoui, was conducted to assess the ways and sources of retrieval of the relevant data. Dr. Maghraoui referred me to Historical Archives and quality published works on Moroccan History which are made available for use by the University Library. An example of such sources is The Historical Dictionary of Morocco by Thomas Kerlin Perk and Aomar Boum.
2.2. SE Process Model

One of this semester’s Capstone Guest Speaker Series talks was about the ways of generating innovative ideas using Design Thinking. One thing that the guest speaker said and that stayed with me was to always think/treat everything as a prototype; basically, to always make room for potential improvement. For the context of this project, I chose to work using the Software Engineering Prototyping process model. Simply put, this model enables us to engineer a prototype based on a set of understood requirements. The idea is that a project never starts with a comprehensive, well formulated and complete set of requirements; and the software engineering team sometimes can lack the means to fully understand the client’s needs. Therefore, to better understand and visualize the requirements, a first, second, and sometimes more prototypes are first made and validated by the client, and after feedback is acquired, the design and the subsequent prototype are adjusted until the client is fully satisfied with the result. For the context of this mobile application, in addition to the previously stated objectives, the goal is also to make room for future expansion [15].

![Figure 3.2.1]

Figure 3.2.1
2.3. Project Management Plan

The following is the project time management planner:

<table>
<thead>
<tr>
<th>Week</th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>• Project selection</td>
</tr>
<tr>
<td>2</td>
<td>• Initial Specification.</td>
</tr>
<tr>
<td>3</td>
<td>• Feasibility Study and Analysis</td>
</tr>
<tr>
<td>4-7</td>
<td>• Requirement Specification</td>
</tr>
<tr>
<td></td>
<td>• Interim Report submission</td>
</tr>
<tr>
<td>8-12</td>
<td>• Design and Implementation start</td>
</tr>
<tr>
<td>12-13</td>
<td>• Implementation and Testing</td>
</tr>
<tr>
<td>14</td>
<td>• Final Report submission and Project Defense</td>
</tr>
<tr>
<td>15</td>
<td>• Updated Final Report submission</td>
</tr>
</tbody>
</table>

3. Project Requirement Specification

3.1. Functional Requirement

- **Search/Determine nearby sites:**
  - The user should be able to determine a search perimeter and should be given/shown the list of nearby sites within this perimeter on the map.

- **Select and View Site Information:**
  - The users should be able to select the desired site (by clicking on a flag in the map or icon in a list) and be redirected to an activity where the site related information (pictures, Description and statistics) are shown.

- **Register/Login/Authentication:**
The users can choose to Register or Login (if previously registered) using their email and password or by authentication using a third party website like Facebook or Google+.

- **Get_Notification(Boolean):**
  - The users should be able to enable or disable the notification center, and if enabled and the app is operating in the background the users should be able to get notification on historical sites that are within their search perimeter.

- **Log off:**
  - Having previously authenticated themselves, the users can choose to log off of the app.

- **Unregister/Unsubscribe:**
  - Registered users can choose to unsubscribe from the app. By doing so, their records are deleted from the database: Wish-list, Favorites, Visited-Places...etc.

- **Rate:**
  - The Registered users can evaluate a site by rating from one to five stars

- **Bookmark:**
  - Registered users will be able to bookmark a site as either Favorite, Visited, or WishList (i.e. a site they would like to visit).

- **Share:**
  - The users will be able to share a site on Facebook, Twitter...etc.

- **View_Bookmarks:**
  - The users will be able to view their lists of Favorite_List, Wish_list, and visited represented using different flags or icons, either on a list-like format or on a map.
3.1.1. Use Case Diagram
The functional requirements mentioned above translate into the following use case diagram

![Use Case Diagram](image)

Figure 4.1.1.1

3.2. Non-Functional Requirement
The app will need to be compliant with the following set of non-functional requirements.

**Performance:** The app should ensure an optimized performance in terms of response and processing time, utilization and throughput.
**Usability:** The app should be user friendly and should provide the user with an enjoyable, trouble free and easy to master user experience.

**Availability:** The app should ensure that the information is available at all times even when the app is offline.

**Scalability:** The app should be scalable and allows for growth without compromising the performance, reliability and availability requirements of the app.

**Maintainability & Extensibility:** The app should allow for ease in maintainability and future growth in functionalities with minimum effort.

**Reliability:** The app should ensure a high degree of reliability by insuring that the service is delivered whenever required with no occurrence of errors or system failures.

**Security & Privacy:** The app should allow for security of data (Confidentiality, Availability and Integrity).

**Other:**

⇒ UI should be compliant with the google and android Material design principles and guidelines (app defined gestures, swipes, left aligned header on android…)

⇒ Time constraint: The app should be ready for release by the due date previously stated.

### 3.3. Database Analysis:

The data that this app will be working with is represented by two entities, one being the user and the other is, the historical site. The following is a detailed description of the two entities and their related attributes.

- **Historical Site:** This entity is the main entity of our system. We mean by a Historical Site any official site with cultural, political, social, military historical significance and heritage. This description could include a variety of location types ranging from a building, site, landscape or structure with a local or global importance. This entity has the following attributes:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>This is the unique identifier that distinguishes the instances from each other within the database.</td>
</tr>
<tr>
<td>Title</td>
<td>This is the descriptive title of the Historical Site</td>
</tr>
<tr>
<td>Type</td>
<td>This attribute describes the type of the historical site, we could have the following types for instance: Archeological Site, Monument, Battle Location, Historical City…</td>
</tr>
<tr>
<td>Location</td>
<td>This attribute represents the location as geolocation coordinates or address of the Historical Site. This is how we will be able to know where the site is on the map and whether it is in proximity of our geolocation.</td>
</tr>
<tr>
<td>Descriptions</td>
<td>This attribute represents the textual description of the historical site, it would consist of historical background description which could include dates, period areas that are pertinent to it and/or the events surrounding its existence, the country, state and government entity it is currently located in, in addition to the one(s) pertinent to its establishment in the past, information about its distinctions if any (pertaining to Global or National Heritage for instance)… the description content can vary based on the different types of sites and availability of the information.</td>
</tr>
<tr>
<td>Rating</td>
<td>This attribute represents the rating of the site from one to five stars as evaluated by the users.</td>
</tr>
<tr>
<td>Images</td>
<td>This attribute represents the descriptive image(s) of the historical site</td>
</tr>
<tr>
<td>Tags</td>
<td>This attribute represents labels that would help identify and give more insights or information about the specific site.</td>
</tr>
</tbody>
</table>
**Registered User:** The User Entity is the main actor of our system. This is the entity that will be using and navigating through the app, making use of the app features and accessing its database.

### Table 4.3.2 User attributes and description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>This is the unique identifier that distinguishes the user instances from each other within the database.</td>
</tr>
<tr>
<td>Email Address</td>
<td>This is the unique public email address that the user uses as an identifier to sign in/up to the app and receive a sign-up confirmation, email from the app admin in addition to more emails pertinent to updates, features and relevant information about the app.</td>
</tr>
<tr>
<td>Password</td>
<td>This is the unique password chosen and known only by the user to successfully connect with the app server.</td>
</tr>
<tr>
<td>Favorites</td>
<td>This is a list of bookmarked sites by the user representing the user’s favorite Historical Sites.</td>
</tr>
<tr>
<td>Wish_List</td>
<td>This is a list of bookmarked sites by the user representing the user’s wish list of Historical sites, i.e., the locations that the user would like to visit.</td>
</tr>
<tr>
<td>Visited_Sites</td>
<td>This is a list of bookmarked sites by the user representing the user’s visited list of Historical sites, i.e., the sites that the user has visited before.</td>
</tr>
</tbody>
</table>
3.3.1. Class Diagram

To better illustrate our data analysis, we use the following class diagram:

![Class Diagram for History Around Me App](image)

Figure 4.3.1.1
4. Project Design

4.1. Technology enablers

For the implementation of this mobile application, as previously mentioned, we chose to develop it for the Android operating system and to make it dynamic, i.e. for it to be developed on two fronts: a server side and a client side. The reason behind this choice is pretty obvious as we wanted it to have features such as authentication and have access to a shared dynamic database in addition to the app being as light as possible in terms of storage. Furthermore, as potential future work, we would like to have the option of pursuing the development of the web app along with one for the IOS operating system which will have access to the same database together with the android app.

The following technology enablers were carefully selected to implement both the front and back end.

4.1.1. Client Side:

![Android Studio](image)

**Figure 5.1.1.1**

This IDE was selected to build the front end of the app. **Android studio** is the official IDE for android mobile development launched by Google in 2013. This IDE was the right choice as it was purposely built and tailored for android mobile development in order to make the process faster and simpler for android developers. It is also made open source in addition to other related tools to encourage the development of more and more android apps, and is made simpler to android newcomers to learn especially with the vast android development community which provides thousands of free online tutorials and platforms that provide assistance to developers.
The **Java** programming language (version 8) was chosen to work on the client side of the app. Java is the official programming language for Android development, it is Object Oriented, portable, and secure. Since Android runs on multiple platforms with different configurations, Java was used for its ability to run on a virtual machine that does the translation from the Java byte code to the native code specific to the computer organization and architecture. Plus, Java is a good choice because I became familiar with it as I have worked with it on several class projects throughout my academic years at the university. Furthermore, there is huge open source support with many available tools and libraries to assist developers using it.

**Figure 5.1.1.3**

**XML**, which stands for eXtensible Markup Language, is a markup language just like HTML and was initially designed for storing, describing and transporting data over the Internet. In my project I used XML for the design of the user interface and layout of the activities, in addition to the screen elements they contain.
4.1.2. Server Side:

Firebase is a Google comprehensive BaaS (Backend as a Service) solution and package for mobile and web app-development.

This platform removes much of the complexity of developing dynamic apps that rely on server side and database link and minimizes server side programming to a great extent. It allows developers to monetize their app while focusing on their users’ needs and apps features and not be concerned with the back end development overhead.

Firebase provides many features for the development and testing of apps such as Real Time Database, Authentication, Crash Reporting, Cloud Storage, Hosting, Test Lab, Crashlytic and more. Furthermore, it provides opportunity for growth and engagement of the audience by the mean of the following features: Cloud Messaging, Invites, Remote Config, AddMob and much more. Below is a description of some of Firebase main features and the most pertinent ones to this project.

- **Real Time Database:**

  This is the original and major feature of Firebase which allows to store and synchronize the data among the users, be it android, iOS or Web platforms, in “real time”, i.e. in terms of milliseconds. This is done through a NoSQL Document Database that stores/structures the data as JSON objects (vs the usual tabular format used with SQL relational databases). We could think of it as a Cloud hosted JSON tree in which, whenever data is added, a new node is added to the tree along with an associated key either user defined or auto generated for us using the push() method. The next snapshot is an example of a JSON.

```json
{
    "users": {
        "some-user-id1458": {
            "email": "123@aui.ma",
        },
```
"password": "123456",
"GetNotification": true
},
"some-other-id1884": { ... },
"another-id65498": { ... }
}

Firebase allows for the availability of the data even when offline by persisting the data to the local Database SDK cache. The data is then synchronized once the networks connection is established.

The Database can also be accessed directly from client devices, be it Mobile or Web, without the need of going through a server. Firebase Real-Time offers the needed security and data validations using its security rules which are expression based rules that get executed whenever a read or write operation is performed.

- **Authentication:**

  Firebase authenticates users rather easily and securely. User authentication is a must regardless of whether it is relevant to the app context since it allows apps to id their users, allocates for added security and collects analytics data pertinent to app usages and user engagement.

  Firebase Authentication supports various authentication methods such as Email/Password, Phone, or third party authentication providers such as Facebook, Gmail, Twitter or GitHub.

- **Cloud Messaging:**

  Firebase Cloud Messaging or FCM for short, allows delivering notification or data messages to the app users regarding triggered events or intending information delivery or user re-engagement.

- **Cloud Storage:** Firebase Storage allows storing user generated content safely and secretly

- **Crash Reporting:**

  This feature allows the generation of detailed reports about the errors and bugs in the app and provides insight about ways of fixing them. Find below a screenshot representing an instance crash report for our app.
• **Test Lab:**

This is a cloud-based infrastructure where developers can test their apps across multiple devices and configuration.

Firebase offers many more interesting features; however, we cannot discuss them all. One more thing worth mentioning is that Firebase’s free pricing plan gives access to all features mentioned above and much more, and is restricted to a 100 simultaneous connections to the database, 1 GB of storage, 10 GB of bandwidth per month… these numbers are perfectly suitable for our project. In the event of willing to grow our app, a fee of $25 per month is required which allows for more than 100k users and a great deal of storage, bandwidth and extended functions and features [3].

*Other technology enablers and concepts worth mentioning:*

NoSQL, which stands for Not Only SQL, is a term commonly used to refer to the emerging and new database design approach, initially intended for the modern web scale databases. NoSQL provides an alternative to standard Relational Databases (where data is represented in a tabular format) and hence is described as being non-relational, highly scalable, available and fault tolerant, open source, schema-free and providing support for distributed
databases, large amount of sparse data, and sacrificing consistency for the benefit of performance. It is said to be “eventually consistent” meeting the BASE (Basic, Soft State, and Eventual Consistency) rather than the usual ACID (Atomicity, Consistency, Isolation, Durability) requirements. **NoSQL Databases** are categorized as either **Document**, **Key-Value**, **Columnar** or **Graph** Databases.

- **Key-value Databases**: simple model pairing every key with a value.
- **Columnar Databases**: adjusted to write/read columns of data as opposed to rows.
- **Graph Databases**: stores data as nodes/vertices and edges/links connecting the edges to each other forming a graph. This type of database can support both relational and non-relational databases.

**NoSQL Document Databases**, this is the model most pertinent to our project since Firebase Real Time Database is described as a NoSQL Document Database. This type stores semi-structured data as documents, usually as JSON or XML documents and is mostly tuned to content management and apps data handling. In addition to Firebase Realtime Database, Couchbase Server, MongoDB, and CouchDB are also described as Document Databases.

**JSON**, which stands for JavaScript Object Notation, is a lightweight data-interexchange human readable format responsible of sending data back and forth from/to the server. It is based on a subset of JavaScript programming language, language independent and easily parsed by most languages. JSON is structured as key/value pairs: `{“appName”: “HistoryAroundMe”}` for instance.

**Google Maps API**: Thanks to this API we can have access to Google Maps Web Services and servers and integrate Google Maps in our application making use of the data and many features provided by this service.
4.2. System Architecture

Our project architecture falls under the Server-less architecture using the BAAS (back end as a service) Firebase which takes care of managing server side logic and state. Here Server-less does not mean that the app is not be dependent on a server, but rather means that the app is reliant on third party services commonly known as FaaS (Function as a Service) or BaaS, Firebase in our case. Such services significantly reduce the cost and complexity of developing n-tier architecture apps. In our case, Firebase takes care of processing HTTP requests coming from the client tier regarding the data stored on the Real Time Firebase Database to be then returned and displayed on the client side. It will also take care of ensuring that the authentication is done successfully, either through email and password or through third party authentication services, such as Google, Facebook, etc.

Figure 5.2.1
Enhanced Architecture

The Application adopts a client Server architecture based on the MVC model. The View (i.e. the client), the Model, (i.e. the database side), and the controller, (i.e. the business logic).

Firebase 5.2.2

4.3. Database modeling

Our data model uses a NoSQL Document Database represented/stored in JSON Tree structure format. The data is manipulated through NoSQL queries.

Here is an example of what a JSON Object might look like. Here we give an example for both the Historical Site and User JSON objects.

**Historical Site Entity:** example

```json

//this is how our database is structured with its key being
//“Historical_Sites” and its value is the entire database of sites
//objects. For every internal node, the key is the Site ID and value is
//the site object and its attribute. This is exemplified underneath.
```
"Historical_Sites": {
    "911761c0-d9e5-11e7-9296-cec278b6b50a": {
        "title": "Archaeological Site of Jebel Irhoud",
        "Description": "Jebel Irhoud or Adrar Ighoud (Moroccan Arabic: جبل إيغود, translit. žbəl iɣud pronounced ['dʒabal ʔiːɣuːd]; Berber: ⵏⴰⴷⵔⴰⵔ ⴰⵏⵉⵊⵓⴷ adrar n iɣud) is an archaeological site located just north of the locality known as Tlet Ighoud",
        "type": "Archaeological Site",
        "rating": 4.9,
        "images": {"0": "data:image/png;base64,/9j/4AA..."},
        "location": {"Lat": "33.5 40985", "long": "-5.110342"},
        "tags": {"0": "Archaeological Site", "1": "HomoSapiens"}
    }, // end of first object
    "911000c0-d9e5-11e7-9296-cec278b6b50a": { ... } // second object
}

User Entity:

{
    "users": {
        "some-user-id1458": {
            "email": "123@aui.ma",
            "password": "123456",
            "GetNotification": true,
        }
    }
}
"bookMarkes": [ { "SiteId": "SiteUUID", "type": "Favorite" }, { "SiteId": "SiteUUID", "type": "WishList" }, { "SiteId": "SiteUUID", "type": "Visited" } ],

"some-other-id1884": { ... },
"another-id65498": { ... }

In this case we chose to reference the related data instead of embedding it since the relationship between site and user is many to many. The same way we use a bridge table in the relational databases to transform many-to-many into a one-to-many, we use referencing here to represent the relation between the two entities.

4.4. Activity Diagram

The activity diagram serves in illustrate the dynamic behavior and flow of our app. This diagram represents the flow and transitions of activities by the user or the system, they can be sequential, branched or in parallel.
Figure 5.4.1
4.5. Sequence Diagram

Figure 5.5.1
This diagram allows us to model the flow of messages and events between the different components of our app. In our app, we identify several components. The first being the user/UI which is directly linked to the mobile app component responsible for sending RESTful request to the business side of the app: Firebase Authentication, Firebase Cloud Messaging, Google Maps, and Firebase Database. The flow of messages can clearly be seen in the following Activity Diagram. Each of these components perform a specific task:

- Firebase Authentication: is responsible for authenticating the user
- Firebase Cloud Messaging: is responsible for sending notification messages to users when a new site enters the search perimeter
- Google Maps: is responsible for first locating the user and showing the sites on the map based on their location
- Firebase RealTime Database: is responsible for storing the site objects and their related information.
5. Project Implementation

5.1. Implementation process:

- **Integrating & configuring Google Maps to track location:**

- **Connecting firebase:**

  Connecting and setting up firebase features is made simple by the presence of a firebase assistant component on Android Studio.

  ![Figure 6.1.1](image)

- **Setting up firebase authentication:**

  After connecting to Firebase, we need to add the following dependencies on the project Gradle level:

  classpath 'com.google.gms:google-services:3.1.1'

  and these to the App Level Gradle
compile 'com.google.firebase:firebase-auth:11.6.0'
compile 'com.google.firebase:firebase-core:11.6.0'

then add the firebase plugin

apply plugin: 'com.google.gms.google-services'

Once this is done, we start working on registering new users: creating a variable mAuth, instantiating it, then creating method registerUser() and making sure to handle basic exceptions like wrong email format, etc. in addition to the user already being registered exception. For the registration to be successful, enabling email, Gmail and Facebook authentication in the Firebase Console web page was necessary, in addition to enabling The Identity ToolKit API on the Google APIs web Page.

Figure 6.1.2

- **Setting up firebase real-time database:**

Since Firebase is already connected, the first step toward setting up Firebase Real Time Database is to add the following dependency to the App Level Gradle.

'com.google.firebase:firebase-database:11.6.0'

- **Populating the Database:**
Our app will only need to read the instances of the Historical site entity and instantiate the user entity when creating a new user account. Hence, since our app does not need to write the historical site data, we will be doing it manually on the Firebase Dashboard Real-time Database web interface. Here is a screenshot of our test database on the Firebase web UI.

Figure 6.1.3

Writing manually to the Firebase Database is done by auto generating a set of unique user identifiers UUIDs using an online UUID generator to be used as object IDs on the database. We need to do this first because firebase database does not generate UUID for manual data entry.
Images for each site object are stored as base64 binary to text encoding scheme.

- **Reading from the Database**

  To read from the database, we need to retrieve an instance of the database and create a reference to the location we want to read from using the following two lines of code, making sure to pass as a parameter the key to the list of sites:

  ```java
  setContentView(R.layout.activity_maps);
  //Firebase DB: Retrieve an instance of the DB using getInstance() and reference the location you
  FirebaseDatabase database = FirebaseDatabase.getInstance();
  DatabaseReference myRef = database.getReference("historicalSites");
  ```

  To read from the Firebase database we need to first create a class Site with the following attributes:

  ```java
  import java.util.ArrayList;
  
  /**
   * Created by Ommaya El Jamaly on 11/20/2017.
   */
  
  public class Site {
    String ID, title, description, type;
    ArrayList<String> tags, images;
    Double rating, locationLat, locationLong;
    
    public Site(){
    }
  }
  ```

  After we create an ArrayList of Sites (ArrayList<Site> sites), we instantiate it by reading the data from Firebase DB using the following piece of code:
We fill the ArrayList sites by iterating over the dataSnapshot (which contains the data as a data snapshot object along with the methods to extract them). The children nodes are fetched using the following snapshot method calls: site.child("attributeName").getValue()

We use the debugger tool to verify that data read is successful. The following screenshot shows that fetching data from the database was successful.
• **Writing to the Database**: when the user will click on the bookmark button, a child node is added to the user object.

• **App Screen Shots:**

![App Screen Shots](image-url)
• More on the implementation in addition to the Screen shots of the app will be included in the Updated Final Report.

6. Project Testing and Validation

Project testing and validation is the fourth step of the software engineering process. The Testing is conducted during or after the implementation in order to ascertain that the product is compliant with the client’s requirements and modeling specifications initially stipulated in the analysis and design phase, i.e. whether or not the product, the History Around Me android mobile application in our case, satisfies and meets the needs, requirements and design initially stipulated. The testing process for our app comprised Unite testing through what is called White Box testing. This is achieved by test targeting specific and small portions of code such as individual classes or functions, for instance when implementing and configuring the authentication many unit tests were performed on functions as they were added to the code, such as entering the wrong email format. The Integration Testing is done by verifying and testing the interaction between the different components of our app as they were integrated in addition to the data flow between them. For instance, in the case of authentication, testing was performed on the registration module and a verification was performed on the server side to check if the user account was actually created. Many errors and system failures occurred during this step and are being dealt with. Concerning System Testing, this is conducted by testing the system as a whole to ensure that the requirements, both functional and non-functional are met. The final step of the testing phase, which represents the Acceptance Testing will be conducted in the presence of Dr. Rachidi a week after the capstone project defense as Dr. Rachidi kindly granted me a week extension to work on debugging the code and finalizing the remaining functionalities.
7. Steeple Analysis

- Social and Cultural: The App will have an added value on the social and cultural context since it will be adding to the collective Moroccan knowledge and historical awareness. It will also add to the popularity of historical sites and facilitate their access which will somewhat contribute to the local tourism revenue of the areas (villages and small cities) surrounding the sites.

- Technical: The app will make use of current and available mobile technology making it easy for users to have access to the information with just a click.

- Economical: The app will have a positive impact on the economy by means of boosting the tourism sector through shedding light on Moroccan Historical sites and their surroundings.

- Environmental: The app does not have any environmental implication be it positive or negative.

- Political: This app does not hold any political implication.

- Legal and Ethical: The app does not have any legal or ethical implication and will not be violating any legal or ethical aspect.
8. Conclusion

The engineering of this project is a requirement under the context of the Capstone Design course and for its completion many steps and actions were undertaken. First, the nature and topic of the project were to be selected and it was thanks to Dr. Rachidi’s precious suggestion of a mobile app that would provide the users with information about historical sites surrounding them. Once the topic in mind, the project was initiated and an analysis study was performed to fully understand the requirements of the app. After the analysis phase came the design phase where several modeling techniques were followed to better emphasize the understanding of the requirements and prepare the ground for the implementation of our app. The implementation phase was initiated and carried on with the requirement in mind and following the blueprints set up during the design phase, whereas the testing and debugging/maintenance of the app was performed in parallel throughout the implementation progress.

To conclude, this project provided me with an opportunity to put into practice many skills, practices and concepts I acquired throughout my academic years at AUI. From the programming classes such as C and Java, to the more conceptual ones such as Software Engineering, Database Design, Computer Networks and Cryptography Systems, therefore providing me with an opportunity to grow as a future software developer and engineer.
9. Future work

Unfortunately, due to time constraints, I was not able to arrive to the result we had envisioned for this project. However, this does not mean that the progress of this project is over, there are still several aspects and functionalities to include in order for this app to take the shape we intended for it. This continuous progress will go into the future work category and will include the following actions:

- Configure Google and Facebook Sign in Methods
- Expand the Database: Adding increasingly more data to our database
- Expand App coverage: by expanding the database, we allow for extended coverage across the country.
- Admin Platform: The Management and administration of the app and the Historical Sites are to be performed not by the developer but by the admin who will be responsible of performing the following tasks (among others) and therefore will need an administration platform to do it. Most probably it will be either a web or desktop app linked to firebase.
  1. Add Site: instantiate sites objects by filling their related attributes such as the name, description and coordinates.
  2. Remove Site: The admin will be able to remove sites for reasons such as non-existence of the site, erroneous information, etc.
  3. Update Site: the admin will be able to update related attributes like update the rating, or the description.
  4. Send Notification.

The future work is not restricted to these actions, as we know a software lifespan is a cycle of continuous refinement and prototyping into improved products and so will be the case for this app.
10. References


10. “System Properties Comparison Firebase Realtime Database vs. GraphDB.” *Firebase Realtime Database vs. GraphDB Comparison*, db-engines.com/en/system/Firebase Realtime Database%3BGraphDB.


