Special Deeds

Internal Management, Reporting, and Communication Solution for an NGO

Younes Elhjouji

Supervised by: Dr. Tajje-eddine Rachidi

School of Science and Engineering
Student Statement

I, Younes ElHjouji, assert that I have applied ethics to the design process and in the selection of the final proposed design. I also confirm that I have held the safety of the public to be paramount and has addressed this in the presented design wherever may be applicable.

Younes Elhjouji

Dr. Tajje-eddine Rachidi
SPECIAL DEEDS

ACKNOWLEDGMENTS:

I would like to thank my supervisor, Dr. Tajje-eddine Rachidi. His guidance and advice throughout the semester was essential for the progress and the completion of this project. He has provided important feedback and insight which are central to the strengths of this project.

I would also like to thank Mr. Mohamed Amin, the president of the Federation of Associations for People with Special Needs in Chtouka Ait Baha. He has been patient and helpful and has given me access to the organization’s current data. His eagerness and passion for automation is why this project was possible.

Moreover, I want to thank the employees and volunteers at the Federation for their patience and for their energy. They introduced and familiarized me with the organization’s operations and detailed to me the problems they face and the solutions they would prefer.

I would like to thank my fellow students Hamza Touhs and Reda Herradi for graciously sharing their expertise with backend and frontend development respectively. Many hours were saved during the development of the system thanks to their help integrating best practices and debugging errors along the way.
SPECIAL DEEDS

CONTENTS:

Introduction 10
   Introduction of the NGO 10
   Introduction of the Problem 11
   Financing and Promotion 11
   Internal Management 12
   Reporting and Communication 12

Feasibility Study 13
   With regards to time 13
   With regards to technologies 13

Requirements Engineering 14
   Elicitation 14
   Non-Functional Requirements 14
   Functional Requirements 16
      Web App 16
      Website 18
      Mobile App 19
         Employees 19
         Beneficiaries 20
   Use Case Example 21

Design 21
   Technology Enablers 22
      Development Stack 22
         Comparative Study 22
            LAMP 22
            MEAN 23
            MERN 24
   Chosen Alternatives 25
      Website 25
      Web App 25
      Mobile App 26
   Class Diagram 26
   Major Technical Considerations 28
      Database 28
      Backend 28
SPECIAL DEEDS

Promises 29
Frontend 30
Component Model 30
Additional Softwares and Libraries used 31
Softwares 31
Atom 31
Robo3T 33
Postman 34
Libraries 34
Axios 35
System Architectures 35
Methodology 36
Software Engineering Model 36
Generic Functions 37
Language Integration 38

Implementation 39
Database 39
Server and Error Handling 40
Client 43
Interface 45
Introduction and Promotion 46
Management 47

Requirement Validation 55
Requirement Validation Approaches 55
Introduction and Promotion 55
Internal Management 55
Reporting and Communication 55
Requirement Validation Table 56

Testing 57
White Box Testing 57
Results and Takeaways 58

Steeple Analysis 58
Social 58
Technological 59
SPECIAL DEEDS

- Economic 59
- Ethical 59
- Political 59
- Legal 59
- Environmental 60

Conclusion 60

Future Work 60
- Handling Donations 60
- Deployment 61
- Integrating Arabic 61

References 62
# SPECIAL DEEDS

## TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>Part of the NGO’s staff and beneficiaries in the Biougra center</td>
<td>13</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Use case for the reporting system</td>
<td>23</td>
</tr>
<tr>
<td>4.1.1</td>
<td>LAMP Development Stack</td>
<td>24</td>
</tr>
<tr>
<td>4.1.2</td>
<td>MEAN Development Stack</td>
<td>25</td>
</tr>
<tr>
<td>4.1.3</td>
<td>MERN Development Stack</td>
<td>26</td>
</tr>
<tr>
<td>4.1.4</td>
<td>React Native flyer showing its cross-platform nature</td>
<td>28</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Class Diagram</td>
<td>30</td>
</tr>
<tr>
<td>4.1.6</td>
<td>MongoDB’s logo</td>
<td>30</td>
</tr>
<tr>
<td>4.1.7</td>
<td>NodeJS logo</td>
<td>31</td>
</tr>
<tr>
<td>4.1.8</td>
<td>Promises flowchart</td>
<td>32</td>
</tr>
<tr>
<td>4.1.9</td>
<td>React’s logo</td>
<td>32</td>
</tr>
<tr>
<td>4.1.10</td>
<td>React’s component model</td>
<td>33</td>
</tr>
<tr>
<td>4.1.11</td>
<td>Atom’s logo</td>
<td>34</td>
</tr>
<tr>
<td>4.1.12</td>
<td>Opening both server and client in atom</td>
<td>35</td>
</tr>
<tr>
<td>4.1.13</td>
<td>Using Robo3t to visualize the database</td>
<td>36</td>
</tr>
<tr>
<td>4.1.14</td>
<td>Making a request using Postman</td>
<td>36</td>
</tr>
<tr>
<td>4.1.15</td>
<td>Example of using axios to link frontend with backend</td>
<td>37</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Client-server architecture diagram</td>
<td>38</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Generic Popup component which is used by many other components</td>
<td>39</td>
</tr>
<tr>
<td>4.3.2</td>
<td>The file containing language options</td>
<td>39</td>
</tr>
<tr>
<td>5.4.1.1</td>
<td>This image shows the gallery and the introductory section</td>
<td>47</td>
</tr>
<tr>
<td>5.4.1.2</td>
<td>This image shows the sidebar used for navigation</td>
<td>48</td>
</tr>
<tr>
<td>5.4.1.3</td>
<td>The lower section of the web page provides contacts and donation link</td>
<td>48</td>
</tr>
<tr>
<td>5.4.2.1</td>
<td>Sign up page</td>
<td>49</td>
</tr>
</tbody>
</table>
SPECIAL DEEDS

Figure 5.4.2.2. Username alone is not enough 49
Figure 5.4.2.3. We try with the wrong password 50
Figure 5.4.2.4. Correct password 50
Figure 5.4.2.5. Successfully logged in, viewing table 51
Figure 5.4.2.6. Adding new beneficiary 51
Figure 5.4.2.7. New Beneficiary successfully added 52
Figure 5.4.2.8. Hovering over delete 52
Figure 5.4.2.9. After clicking on delete, beneficiary is deleted 53
Figure 5.4.2.10. Updating added beneficiary 53
Figure 5.4.2.11. Changing last name to match last name of legal guardian 54
Figure 5.4.2.12. Last name updated successfully 54
Figure 5.4.2.13. Viewing the services possible 55
Figure 5.4.2.14. Checking the offerings of services 55
Figure 5.4.2.15. More information about one of the offerings 56
SPECIAL DEEDS

Abstract

This project aims to provide an integrated solution for the Federation of Associations for People with Special Needs. The solution is composed of three main parts: a static webpage, a webapp, and a mobile app. The webpage is a promotional and fund-raising webpage. The web app is an app for internal management of the NGO by the director. The mobile App is for reporting and communication with the beneficiaries of the NGO. The project uses a MERN development stack (MongoDB, Express, ReactJS, NodeJS). The system is built according to the client-server architecture and following the agile development model. The project was tested throughout development using the white box testing method. The following report goes through the steps of this project from the problem definition to the implementation.

Keywords: Web Development, Mobile App Development, Automation
1. Introduction

1.1. Introduction of the NGO

The NGO that I worked with for this project is the Federation of Associations for People with Special Needs. In the remainder of this report, the name will be abbreviated as FAPSN. The NGO conducts many operations to help people with special needs including giving haircuts, helping with housing and healthcare, and teaching and providing care for children with disabilities. The latter is their main activity, they provide education, diagnosis, and therapy for children with disabilities. For this reason, they employ many caretakers and teachers in order to integrate the children and to help their parents.

The Chtouka Ait Baha region where the NGO operates lacks in education and integration of children with special needs and the NGO attempts to fill that gap both to integrate the children, as well as to assist their parents and free their time for their jobs. This is especially important because most of the beneficiary families are low income families who would otherwise have to choose between the wellbeing of their children and the livelihood of the family. The children are put in classes according to their disabilities and caretakers assist them accordingly. The NGO has two centers: one in Biougra and one in Sidi Bibi, both in the Chtouka Ait Baha region. By the end of my time volunteering with the NGO, the director asked if I could help automate their operations to solve some of the problems they have. Those are listed in the next section.
1.2. Introduction of the Problem

I volunteered with the NGO as part of my CIP hours required by Al Akhawayn University. During my time volunteering at the organization, I got familiarized with their operations but also with some of the problems they are facing. These problems are in three main categories:

1.2.1. Financing and Promotion

The organization naturally has many expenses, including the pay of their employees, the equipment needed for the care they provide, and the activities they perform (competitions, trips, etc). However, they do not have a guaranteed stable source of income. Their funding comes mainly from local and regional councils as well as from donations by other organizations. Both of these sources of income are not stable, since the councils change their budget frequently and the donations mostly come in one-time donations rather than streams of funding. For this reason, the NGO often struggles financially and is under-staffed due to budgetary concerns.
SPECIAL DEEDS

For now, the organization has only a Facebook page for promotion and only get personalized donations without any one official channel for donations. They want to change that and have an official website in French and Arabic since many of their donations come from European organizations.

1.2.2. Internal Management

Working with children with special needs requires follow-up and personalized care. This is almost impossible for the NGO since they are understaffed and the information about beneficiaries gets drowned out in files of paperwork. Retrieving the history of any single student or deciding which class they should be assigned to becomes a very difficult task. Moreover, the NGO managers struggle with assigning caretakers to sections and keeping track of which employees are currently doing which job. All of this leads to overhead in management because of the reliance on paperwork.

1.2.3. Reporting and Communication

Other than internal reports and documents, there are reports that the NGO attempts to make available to the legal guardians of their beneficiaries. However, due to restrictions that paperwork imposes on the NGO, parents have to be physically present at one of the centers to follow up with their children’s progress, diagnoses, etc. This is difficult for many parents who live far away and ones with weaker financial standings. This also brings more overhead to the organization since employees have to physically handle the reports and make them available for the parents.

In order to solve these problems, the director of the NGO and I agreed that the organization needs three things: a static webpage, a web app, and a mobile app.

The static webpage will serve for the promotion of the website, and will serve as the bridge the organization has with outsiders. It provides informations about the organization, their operations and their contact information and provides a donation channel for people and organizations willing to donate.

The web app will serve as a facilitator for internal management allowing the manager(s) to manage the beneficiaries, employees, legal guardians, and services provided with the
organization. It automates the operations and reduces the need for conducting management using paperwork.

The mobile app will be used as a way of reporting by the employees and as a means of communication between the caretakers and the legal guardians. It is connected to the same backend and database as the web app but it offers different levels of access and allows for specific services: mainly reporting and communication.

2. Feasibility Study

2.1. With regards to time

The time frame for this project is the current semester. The project should be completed before the 22nd of April. This time frame is sufficient to develop such a product. However, in order to ensure timeliness, implementation must be started as soon as possible, and the diaries must reflect progress that is made every week.

2.2. With regards to technologies

The NGO that I am working with asked for two things:

A website to help with the NGO’s fundraising and management.

A mobile app to help with reporting and communication with the beneficiaries’ parents.

For the website, I will be working with the following languages:

- html
- css
- javascript

In addition, I will be using the following frameworks:

- React
- NodeJS
- Bootstrap

For the Mobile App, I aim to make it usable for on both Android OS and iOS phones. To achieve this, I will use react native which allow for cross-platform development for both Android and iOS.

The languages used for the mobile app will be:
SPECIAL DEEDS

- SQL
- javascript
- XML

For the frameworks that will be used, those will include:
- React Native
- NodeJS

Of these languages and frameworks I am already familiar with javascript, SQL, html, css, and bootstrap. I have to familiarize myself with XML, react, react native, and NodeJS. This is feasible within the time frame of the capstone project.

More information about the technology enablers used and the reason for the choice of technologies are in the following sections.

3. Requirements Engineering

3.1. Elicitation

Elicitation for this project mainly consisted of communication with the director both in person and through the internet as well as access to the forms and documents currently in use by the organization. This process was relatively smooth thanks to the director’s commitment to automating the organization as well as the help of employees who also answered questions about the functionalities they want to have in the mobile app and how they need them to be provided.

The elicitation also included my having access to the digital data already in use by the company which consisted of simple databases that needed to be improved and normalized before they were integrated into the products.

3.2. Non-Functional Requirements

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Security</td>
<td>Encryption</td>
<td>The Webpage shall contain the donation channel, and the database for the web</td>
</tr>
<tr>
<td>SPECIAL DEEDS</td>
<td>Access Control</td>
<td>Secure Donations</td>
<td>app and mobile app shall contain sensitive information about the company’s employees such as salaries etc as well as about the beneficiaries such as diagnoses. For this reason, all the sensitive information shall be secure and access to it shall be adequately controlled in order to prevent unauthorized access to it</td>
</tr>
<tr>
<td>2</td>
<td>Usability</td>
<td>Cross-platform support</td>
<td>The mobile app shall be usable for both android and iOS users. It shall also be usable in older phones and devices.</td>
</tr>
<tr>
<td>3</td>
<td>Deployment</td>
<td>Deployment options</td>
<td>All three components of the digital solution shall be deployed and accessible by the 1st of June. The NGO prefers that deployment be free. If free deployment through a service provided for NGOs for free, the organization is ready to cover the expenses of deployment.</td>
</tr>
</tbody>
</table>

### 3.3. Functional Requirements

#### 3.3.1. Web App

The website must include management functionalities, those are detailed below:
<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Language</td>
<td>Arabic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>French</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Language switching</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The director wants the promotion webpage to be in Arabic and French to reach the organization’s donation sources and to be readable by both local users and international prospective partners. As for the web app and the mobile app, the director wants them to be in Arabic as it is the language that employees and legal guardians are most familiar with.</td>
</tr>
<tr>
<td>5</td>
<td>Manage Beneficiaries</td>
<td>- Add Beneficiary (Enrollment)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update Beneficiary Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- View Beneficiary(ies)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Delete Beneficiary (Exit)</td>
</tr>
<tr>
<td>6</td>
<td>Manage Employees</td>
<td>- Add Employee (Hiring)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update Employee Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- View Employee(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Delete Employee</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Leave)</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
</tbody>
</table>
| 7 | Manage Legal Guardians | - Add Legal Guardian  
- Update Legal Guardian Information  
- View Legal Guardian(s)  
- Delete Legal Guardian |
| 8 | Disability | - Add Disability  
- Update Disability Information  
- View Supported Disabilities  
- Delete Disability |
| 9 | Diagnoses | Diagnoses are made by the psychiatrist and kept track of in the app. A diagnosis associates a beneficiary with a disability. The functionalities linked with diagnoses are:  
- Add Diagnosis  
  (Diagnose beneficiary)  
- View a beneficiary’s diagnoses  
- View beneficiaries with a similar diagnosis  
- Update a diagnosis |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>- Delete a diagnosis</th>
</tr>
</thead>
</table>
| 10 | Service | The services that the NGO offers. A service is linked with a disability or a number of disabilities and has its own description and information. e.g. (Physical therapy, pronunciation aid, sign language education…). The functionalities linked with a service are:  
- Add a Service  
- Link Service to Disability  
- View Beneficiaries Benefiting from Service  
- Update Service  
- View Service information  
- Delete Service (if no longer offered) |
| 11 | Offering | An offering of a service, since services can be offered multiple times or by different caretakers. Offering functionalities are: |
| 12 | Reports | Reports are filled by employees on beneficiaries. They are filled through the mobile app, but the Web App shall offer the following functionalities for the management of the reports:
  - View reports
  - Delete report
The update functionality is not provided for management since only the employee who issued a report shall have the ability to update it. |
3.3.2. **Website**

The website is mainly for promotional and financing reasons and therefore it has fewer functionalities than the Web or mobile App.

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Information Display</td>
<td>The Website must display information about the organization. Users if the website can view the information which shall be presented in an accessible and multilingual way in accordance with the non-functional requirements.</td>
</tr>
<tr>
<td>14</td>
<td>Images Display</td>
<td>In addition to information in text form, the website must additionally display images of the organization’s activities and fliers. The images shall be available in galerie format. Users of the website shall have the availability to scroll through images.</td>
</tr>
<tr>
<td>15</td>
<td>Contact Links</td>
<td>The website shall link to the contacts of the organization through links and information (phone numbers etc)</td>
</tr>
<tr>
<td>16</td>
<td>Donations</td>
<td>The website shall have a donation functionality which allows users to donate to the NGO.</td>
</tr>
</tbody>
</table>

3.3.3. **Mobile App**

The mobile app shall have two uses, one for employees of the organization and another for the Legal Guardians of the beneficiaries, the requirements for each are listed below.
### 3.3.3.1. Employees

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
</table>
| 17     | Reporting   | Employees shall access the reporting history of beneficiaries under their care and add new reports. The functionalities are:  
- Add report  
- View report history  
- Update report  
- Delete Report |
| 18     | Communication | Employees must have the ability to communicate with the legal guardians of beneficiaries in order to communicate to them about their children. The functionalities are:  
- Send message  
- View conversation |
| 19     | Viewing Privileges | Employees shall have access to some information about the organization including the information of the disabilities served, the services provided, and information about beneficiaries under their care |

### 3.3.3.2. Beneficiaries

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Reporting</td>
<td>Legal Guardians shall have viewing access to the reports of their children therefore, they must be able to:</td>
</tr>
</tbody>
</table>
### Special Deeds

| 21 | Communication          | Like employees, legal guardians shall be able to:                                                                 |
|    |                        | - Send message                                                                                                    |
|    |                        | - View conversation                                                                                                |

| 22 | Viewing Privileges     | Beneficiaries shall be able to:                                                                                   |
|    |                        | - View their children’s information                                                                              |
|    |                        | - View their children’s diagnoses                                                                                 |
|    |                        | - View offerings their children are enrolled in                                                                  |
|    |                        | - View services provided by the organization                                                                    |
|    |                        | - View which employees are supervising their children                                                             |

#### 3.4. Use Case Example

The following presents the use case diagram for a the reporting system in the app. The reporting system was used as an example because it used by all users of the system and it therefore shows how one module can have a complex use case.
Figure 3.4.1. Use case for the reporting system
4. Design

This section provides information about the next step of software engineering: the design step. It shows and justifies the choices of technology enablers used, details the system architecture, and provides examples from the code. It also details the methodology choices made to facilitate the development of the project.

4.1. Technology Enablers

The technology enablers used for this project include not only the chosen development languages and frameworks but also the softwares used to facilitate development. Examples of these softwares will also be provided in the following.

4.1.1. Development Stack

The choice of a development stack is essential to the design of the project, different development stacks all have strengths and weaknesses. Therefore a comparative study is important in order to determine which development stack is most efficient for each component of the project (Website, Web app, Mobile app). Due to the budgetary limitation, any development stack that is not fully open source could not be considered for this project. The development stacks compared are therefore all open-source.

The following section provides the comparative study conducted for this project.

4.1.1.1. Comparative Study

4.1.1.1.1. LAMP
## SPECIAL DEEDS

<table>
<thead>
<tr>
<th>Components</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux (Optional)</td>
<td>Most conventional stack, therefore plenty of documentation and information is available. [1]</td>
<td>Web App only</td>
</tr>
<tr>
<td>Apache (Web Server)</td>
<td></td>
<td>Not easily adopted with mobile applications. [1]</td>
</tr>
<tr>
<td>MySQL (Database)</td>
<td>Large Community. [1]</td>
<td>Relatively old approaches and techniques. E.g. relational database</td>
</tr>
<tr>
<td>PHP (Backend)</td>
<td>Platform Independent. [1]</td>
<td>Requires skills in all the languages used</td>
</tr>
</tbody>
</table>

### 4.1.1.1.2. MEAN

![MEAN Development Stack](image)

Figure 4.1.2. MEAN Development Stack

<table>
<thead>
<tr>
<th>Components</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB (Database)</td>
<td>Uses Javascript for both backend and frontend. [2]</td>
<td>Does not expand easily to mobile app development</td>
</tr>
<tr>
<td>Express (NodeJS framework)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular (Frontend)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NodeJS (Backend) | Integrates newer best practices. [2]  
| Component-based frontend  
| Asynchronous event-based backend. [4]  
| Offers access to many frameworks built on top of NodeJs and AngularJS. [2]  
| Requires learning Express, NodeJS, and Angular

4.1.1.3. MERN

Figure 4.1.3. MERN Development Stack

MERN and MEAN only differ in their frontend development technology, MERN uses ReactJS rather than AngularJS, therefore the difference between them is small but it is also an essential one for this project

<table>
<thead>
<tr>
<th>Components</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB (Database)</td>
<td>Uses Javascript for both backend and frontend [2]</td>
<td>Requires learning Express, NodeJS, and ReactJS</td>
</tr>
<tr>
<td>Express (NodeJS framework)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.1.1.2. Chosen Alternatives

#### 4.1.1.2.1. Website

The website will be a static website, therefore it shall be developed using standalone **ReactJS**. This choice was made since ReactJS is also used for the web app and the mobile app and therefore it is easier and faster to develop the website in ReactJS as well.

#### 4.1.1.2.2. Web App

For the Web App, the choice made is to go with **MERN** as a development stack due to the strengths stated above. Especially its adaptability to mobile app development. [2][3]
4.1.2.3. Mobile App

MERN was also chosen for the mobile app since ReactJS has a mobile application parallel: React Native.

![React Native flyer showing its cross-platform nature](image)

React Native is a good fit for this project because, not only share most of its structure with ReactJS, but it also allows for simultaneous development for both android and iOS. This means that the skills learned in ReactJs are transferable to React Native and it also means that development time is split in half because of the cross-platform nature of React Native.

4.1.2. Class Diagram

The following class diagram shows the entities used for the project and details their attributes and the functions that can be applied to them.
Figure 4.1.5. Class Diagram
4.1.3. Major Technical Considerations

4.1.3.1. Database

For the database choice, I used MongoDB, which is a noSQL DBMS that is different from the usual relational databases in that it has fewer restrictions on what documents can be added to the database. I made this choice over a relational database because MongoDB provides more freedom to restructure and change the database during development than is usually offered by relational DBMSs. [4]

In addition to MongoDB, and in order to define the structure of my database, I used Mongoose, which is a Javascript framework that works with NodeJs and allows for definition of models (entities) and for the modification of those definitions during development. [4]

4.1.3.2. Backend
For the backend, I use NodeJs which is Javascript based, server-side frameworks that allows for smooth and fast backend development and which comes with many packages that make development easier as well as with npm, the package manager. The main package I worked with on top of node is ExpressJS which makes coding using NodeJS easier and facilitates routing. [4]

4.1.3.2.1. Promises

NodeJS uses an asynchronous approach, this means that I had to work with promises in order to manage the flow of my software. Promises offer a way to continue running code until an asynchronous function is finished running before using the values returned to perform an operation. The way promises work is by asynchronous functions immediately returning promises, which serve as placeholders for the return value and which are “empty” until the asynchronous function is done running, which resolves the promise and allows the code that depends on that promise to be run.
For the frontend development I used ReactJS, which uses the component based model and greatly facilitates designing dynamic web pages and linking them to the backend. ReactJS uses the component model which is briefly explained in the next section.

4.1.3.3.1. Component Model

Figure 4.1.10. React’s component model
SPECIAL DEEDS

ReactJs allows the developer to create Javascript components which are the compiled into DOM components. However, the ReactJS model has some added benefits: Components have states and are instantiated with props. States can be looked at as the local variables of a component and props can be looked at as the parameters passed to that component. [3]

Components also have code that allows them to link to the backed and to change their state. With every change to the state, ReactJs updates the component and re-compiles it into DOM. This greatly increases the efficiency of code as well as it readability since components are more organized than DOM elements. [2]

4.1.4. Additional Softwares and Libraries used

4.1.4.1. Softwares

4.1.4.1.1. Atom

Figure 4.1.11. Atom’s logo

Instead of using different IDEs for the different parts of my project, I opted to use the Atom text editor combined with the Linux terminal for development. That is for the following reasons:

- Atom is a free open-source software: This means that I didn’t incur any expenses related to IDEs
- Atom is lightweight: The development of such a large scale project is hardware-demanding, it is therefore almost impossible to run multiple IDEs as well as the server and the client on my hardware.

- Atom offers many helpful packages: These packages include code-snippets, reformatting of code, and an integrated terminal for easy access. The availability of these packages meant that I had more control over my development environment.

Figure 4.1.12. Opening both server and client in atom
4.1.4.1.2. **Robo3T**

Figure 4.1.13. Using Robo3t to visualize the database

Robo3t is a software that allows for visualization and editing of MongoDB databases. It was greatly helpful for the development of the database and the backend as it allowed me to directly access and modify the database. It takes advantage of MongoDB’s fluidity to offer the user great direct control of their database.
4.1.4.1.3. Postman

Postman is a software that allows for making API calls for testing and development purposes. It allowed me to test whether my API calls were working without having to make API calls from the frontend. In this way, it greatly sped up development.

4.1.4.2. Libraries

This work would not have been possible within the given timeframe without the use of libraries that were built on top of NodeJS and ReactJS. The libraries include Mongoose, Express, Axios, and react-table. The following is an example of the use of one of these libraries.
4.1.4.2.1. Axios

Axios is a Javascript library that can be used on the client side to allow and facilitate API calls to the server. Using axios, API calls can be structured in a straightforward and replicable way. This means that adding headers becomes easier as well as adding params and data to a request.

4.2. System Architectures

The mobile app part of this project and the web app part both make use of the same backend in order to provide their functionalities. Therefore, the System architecture used had to provide a backend that was useable from two separate frontends. The System architecture which I found to best accommodate this need is the client-server architecture.

The client-server architecture allows multiple clients to use the same server. The server shares resources with by responding to requests and sharing resources through responses to requests. The clients in this case do not communicate except for through the server, which is the only entity they share resources with. [5]
Using the client-server architecture allowed me to work on the web app alone first, designing both its frontend and backend. Once I moved on to the mobile app, I did not have to redesign the backend, but I used the same server already in use by the web app.

### 4.3. Methodology

#### 4.3.1. Software Engineering Model

The Software model followed during this project is the **Agile Development Model** and more specifically the feature driven development. Agile development is efficient at responding to unexpected and quick changes. It is also fitting for short development times since it cuts down on time taken for development of a project because it fits the process of making the product to the product itself. [8]

The Agile model is a broad model of software engineering that was developed in the nineties. Therefore, there are many schools of thought within agile development each differing in the details of their approach. The specific agile model followed for the completion of this project is the feature-driven model. [8] This model follows the agile approach of iterative and adaptable
development and adds to it a specific focus on features as the important units of the process. The project is divided into features that have subfeatures. Each feature must not take a long time to implement, otherwise it should be divided into subfeatures. The features are fully made one at a time, or in parallel if they are codependent. [9]

In my project, this approach was extremely helpful, both because it allowed for a feedback loop with the client and because the project had to be built like a pyramid starting with the more basic features — e.g. Adding employees and beneficiaries — to the more complex ones — e.g. Connecting beneficiaries and employees through sections.

4.3.2. Generic Functions

![Image of code](image.png)

Figure 4.3.1. Generic Popup component which is used by many other components

Many aspects of the project were quite repetitive of their nature. For example, almost of the tables in the admin dashboard use similar backend API calls and provide similar CRUD functionalities. Therefore, instead of repeating code for every table. I took advantage of the strengths of ReactJs to design a generic table which takes is the table content as its arguments
SPECIAL DEEDS

and formats it accordingly and then adds the shared functionalities. This approach was used not only with tables but with all the components that were used repeatedly in the project, such as Popups, Forms, and lists.

4.3.3. Access Control

For access control, I have used API calls for:

- Sign in
- Sign out
- Sign up
- Authenticate

Once the user has signed in, the frontend is issued a token that needs to be sent with all future requests to authenticate the user before the response is sent.

I have three access levels in the web app and mobile app: Admin, employee, and legal guardian and they have access to different functionalities as per the functional requirements. Requests contain in them the access level requested and the token of the user making the request. Upon receiving the request, the server authenticates the token and checks if it corresponds to a valid userSession that has the right access level and if it has not expired. On the frontend side, if any illegal request is made (by the user typing out a protected url for example) the client receives an error that shows that an illegal call was attempted. The client reacts by redirecting the user to the home page.
4.3.4. Language Integration

I chose to develop the project in English, and then change it to Arabic for the following reasons:

- Using Arabic in the middle of my code caused formatting issues and was difficult to combine with the latin script of the code.
- As I am working on a capstone in an American university, I wanted to present my demo in English.
- Switching the keyboards and typing in Arabic while coding would have caused overhead in the time it takes to code.

For these reasons, I decided to develop in English but to allow for easy integration of Arabic later on. This was possible by using a language folder which contains a JSON object representing all the text that I have in my three products as variables that have both English and Arabic values.
With this module integrated, instead of directly typing “Welcome to our Website” in my code, I would declare the sentence in my JSON object in both Arabic and English and then reference it in my code as:

```
language.messages.welcomeMessage[lang]
```

The variable ‘lang’ in this case would hold either the value “ar” or “eng” and would be a global variable. Meaning changing the local variable from “eng” to “ar” would change all the texts displayed to the user.

5. Implementation

5.1. Database

This is an example of one of my database models. The models are defined as JSON objects which define the attributes, whether they are required, their type etc.

```javascript
var mongoose = require('mongoose');
const bcrypt = require('bcrypt');
var Schema = mongoose.Schema;

var userSchema = new Schema({
    username: {
        type: String,
        default: ''
    },
    password: {
        type: String,
        default: ''
    },
    email: {
        type: String
    },
    access: {
        type: String,
        default: ''
    },
    creationDate: {
        type: Date,
    }
});
```

This code segment shows how the Mongoose library is used to define a user schema. The schema defines the restrictions placed on the user collection in the database. Then a model is made using that schema and exported to be used in other parts of the code. The model’s utility is that it allows for database operations (find, add, update, delete…) to be performed on the user collection.

5.2. Server and Error Handling

After seeing how a user is defined in the database, this section will show snippets of code from the server that makes use of the user collection through the export. It will show the logic of how API calls are handled in the backend.

Signin gets the username and password of the user attempting to Signin. It sends back an error if one of those is missing. If not, it checks if the code and username is valid, in which case it creates a userSession and returns information about the created userSession as well as a token to be used by the user to verify future requests made by said user. The code for it is as follows:

```javascript
const User = require('./../../models/user');
const UserSession = require('./../../models/userSession');
```
module.exports = (req, res, next) => {

const {body} = req;

let {username, password} = body;

// Error handling
if (!username || !password) {
    return res.send({
        success: false,
        code: 2,
        message: 'Make sure you fill all fields'
    });
}

// Verify that user does not exist
User.find(
    {
        username: username
    }, (err, users) => {
        if (err) {
            return res.send({
                success: false,
                code: 1,
                message: 'Server error'
            });
        }
        else if (users.length != 1) {
            return res.send({
                success: false,
                code: 7,
                message: 'Invalid username or password'
            });
        } else {
            const user = users[0];
            // Authenticate password
            if (!user.validPassword(password)) {
                return res.send({
                    success: false,
                    code: 3,
                    message: 'Password is invalid'
                });
            } else {
                // Authenticate user
                return res.send({
                    success: true,
                    code: 0,
                    message: 'User authenticated'
                });
            }
        }
    });

);
In the top of the shown code, there are two imports (requires) of database models (user and userSession) that are used during the Signin logic.
5.3. Client

Now that we have our user collection in the database, and the Signin logic in the server, the next step is to see how this is all presented to the user and how user input is connected to these components. The following code show the full Signin component in ReactJs.

```jsx
import React from 'react';
import axios from 'axios';
import {withRouter} from 'react-router-dom';
import {setInStorage} from './../utils/storage'

class Signin extends React.Component {
constructor(props){
  super(props);
  this.state = {
    error: "",
    username: "",
    password: "",
    email: ""
  }
  this.handleChange = this.handleChange.bind(this);
  this.handleSubmit = this.handleSubmit.bind(this);
}

handleChange(event){
  var newState = this.state;
  newState[event.target.name] = event.target.value;
  this.setState(newState);
}

handleSubmit(event){
  event.preventDefault();
  axios.post('http://localhost:5000/users/signin',{ 
    username: this.state.username,
    password: this.state.password
  })
  .then((res) => { 
    this.setState(res);
    alert('User logged in successfully');
  })
  .catch((error) => { 
    this.setState({error: error.message});
    alert('Error logging in');
  });
}
}
```
let {data} = res;

if(!data.success){
  this.setState({error: data.message});
}
else{
  this.setState({error: ""});
  setInStorage("token", data.token);
  this.props.history.push("/admin");
}

}).catch((err) => {
  console.error(err)
});

render(){
  return(
    <div >
      <h2>Sign in </h2>
      <p className = 'error'>{this.state.error}</p>
      <form>
        <label className="form-group">Username:
          <input className="form-control"
            type="text"
            name='username'
            value={this.state.username}
            onChange={this.handleChange}/>
        </label>
        <label className="form-group">Password:
          <input className="form-control"
            type="password"
            name='password'
            value={this.state.password}
            onChange={this.handleChange}/>
        </label>
    </form>
  </div>
)
We can see that the Signin component has:

- State: which holds the values of the Signin fields
- Functions: which handle change of value of a field as well as clicking the Signin Button.
- Markup: The return of the component, which defines the Markup of the page and links the values inside the DOMs to the state of the object.

After the signup is done successfully, the user is redirected to the admin page. If the user’s access does not allow them to access the admin page, they will be told that they do not have the right access credentials and redirected to the login page.

5.4. Interface

This section shows the interface of the project and the work and showcases how operations can be done.
5.4.1. Introduction and Promotion

Figure 5.4.1.1. This image shows the gallery and the introductory section

Figure 5.4.1.2. This image shows the sidebar used for navigation
Figure 5.4.1.3. The lower section of the web page provides contacts and donation link

5.4.2. Management

![Sign up form](https://www.youtube.com/watch?v=U0v4ew69m7Q)

Figure 5.4.2.1. Sign up page
Figure 5.4.2.2. Username alone is not enough

Figure 5.4.2.3. We try with the wrong password
Figure 5.4.2.4. Correct password

Figure 5.4.2.5. Successfully logged in, viewing table
Figure 5.4.2.6. Adding new beneficiary

Figure 5.4.2.7. New Beneficiary successfully added
SPECIAL DEEDS

Figure 5.4.2.8. Hovering over delete

Figure 5.4.2.9. After clicking on delete, beneficiary is deleted
SPECIAL DEEDS

Figure 5.4.2.10. Updating added beneficiary

Figure 5.4.2.11. Changing last name to match last name of legal guardian
SPECIAL DEEDS

Figure 5.4.2.12. Last name updated successfully

Figure 5.4.2.13. Viewing the services possible
SPECIAL DEEDS

Figure 5.4.2.14. Checking the offerings of services

Figure 5.4.2.15. More information about one of the offerings, shows links and offers adding or deleting links
6. Requirement Validation

6.1. Requirement Validation Approaches

6.1.1. Introduction and Promotion

For the introduction and promotion website, I based my approach on continuous communication with the client. He would describe to me what the site should look like, I would try to approximate his imagination and then I would show him the project. The client would then give me feedback to improve the website until we could agree that each requirement was fulfilled.

6.1.2. Internal Management

This part of the project had requirements that are pretty standard for a dashboard application. In order to fulfill each requirement, I made sure that:

- The user could make CRUD operations on the required entity
- The links that the entity requires are established and useable

Then I would move on to the next entity. I attempted to start with the entities that do not contain any pointers to other entities before moving to the other entities. Otherwise, I would have had to implement linking mechanisms before I had anything to link to.

I also had to make sure the Internal management part of the project was secure. To this end, I followed cryptography and security best practices like hashing the passwords with a salt before they are saved etc. I also had to implement access control to make sure only the user can get access to the data and functionalities of internal management. I implemented access control in the backend since it might be dangerous to implement it in the frontend.

6.1.3. Reporting and Communication

Reporting and communication are based on the database populated and managed by internal management. Therefore, implementing this part did not require much changes to be done to the backend.
SPECIAL DEEDS

The mobile app allows access to employees and legal guardians and each have different functionalities. Therefore, the task is to create essentially two mobile applications in one. Both Apps are connected to the backend through which they communicate. In accordance with the client-server system architecture, there is no direct connection between the employee app and the beneficiary app.

6.2. Requirement Validation Table

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Fulfilled</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Language</td>
<td>In progress</td>
</tr>
<tr>
<td>2</td>
<td>Security</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Usability</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Deployment</td>
<td>Not yet</td>
</tr>
<tr>
<td>5</td>
<td>Manage Beneficiaries</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Manage Employees</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Manage Legal Guardians</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Manage Disabilities</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Manage Diagnoses</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Manage Services</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Manage Offering</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Manage Reports</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Information Display</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Images Display</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## 7. Testing

### 7.1. White Box Testing

The method used for testing of the project so far is white box testing. This method requires access to the code and knowledge of the implementation choices and techniques. Because I am the sole coder for this project, it is possible for me to go through the code, look for possible errors and legal inputs, test them, and improve my code according to the result.

I also opted to test units at least twice: one when they are standalone units and again after they have been integrated into the project. This is in accordance with white box’s differentiation between unit testing and integration testing. As the system starting coming together, I moved on to system testing, to make sure that the two clients (mobile app and web app) do not cause inconsistencies in the system.
7.2. Results and Takeaways

Testing was not done at the beginning of the project. After a few problems with complex error, I decided to adopt meticulous testing to ensure possible errors are handled before they have a system built upon them and dependant on them. One of the areas where my testing revealed many error was the process of establishing and managing links between entities. As a reaction to this, I updated my database to follow the best practices that best deal with errors. I also built generic linking functions to implement strong error handling once rather than duplicating code multiple times. [10]

Another result of doing testing throughout the project is that it sped up learning and improvement through the development process. As I found and corrected errors, I made sure I did not reproduce the in future steps of development. I estimate that this greatly sped up the development process.

I made sure to also contact the client and preview my progress at every point in order to have the opinion of a stakeholder who might approach the product and UI differently than I do. In doing this, I had to update my interface to better fit the client, which took time but ensured I did not create an unusable product.

8. Steeple Analysis

In this steeple analysis, I will go through the social, technological, economic, environmental, legal, political, and ethical factors that are relevant to this project.

8.1. Social

The project is social by its very nature. It was taken for the purpose of helping a social NGO provide a social service to an underprivileged society. When developing the project, I have
SPECIAL DEEDS

to take into consideration many social aspects, like the languages available in my software and the presentation and how it will be received by the users who are not generally tech savvy.

8.2. Technological

Speaking of technology, we go to the next aspect. Biougra, where the NGO exists, is a poorer area where people have limited access to technology. Smartphones are just becoming the norm and most people do not own personal computers. This means that in order to make the software accessible to as many people as possible, it needs to be available both as a mobile app and as a web app that anyone can access from a library or internet cafe.

8.3. Economic

Part of my project will be a fundraising website to raise funds for the NGO. This has economic relevance since making the website accessible to Western English and French speakers, we can introduce the NGO to a whole new class of donors who are generally richer and who can contribute more to the NGO’s finances.

8.4. Ethical

As a developer of an online mobile and web app, I have to take into consideration the maintenance and upkeep of the server and the database. More importantly, it is my ethical responsibility and obligation to maintain the apps and respond to the NGO for any updates needed in the future. I understand that this might take large portions of my time for years to come but I cannot ethically render the NGO dependant on a system that I do not maintain afterwards. [6]

8.5. Political

The environment of Biougra is very political despite it being a small town. The inhabitants always complain about corruption and lack of responsibility among government workers and even elected representatives. In the past, the NGO suffered problems with funding because they were seen as wasteful by some government official who might have wanted to be bribed to return the funding. My project will only lightly touch on the political side by giving more exposure to the NGO (through the website) and by optimizing their operations, but it is important to consider any possible intersection between the project and the political dimension.
8.6. Legal

I talked with my client to make sure we are not operating outside of the boundaries of what is legal. He assured me that we aren’t and specifies that the NGO strives to uphold best practices and to serve as an example of an NGO that follows the laws to the letter. This is in big part because the NGO workers and volunteers take their work very seriously and because Moroccan NGOs risk losing funding if they participate in questionable practices.

8.7. Environmental

When I was volunteering with the NGO, I saw that they used a lot of paper to operate. I saw that much of that paper ended up being burned, lost, or thrown in trash. Leading to both loss of data as well as polluting the environment. Providing an automated alternative will not only speed up operations, but also help the environment as it will reduce the pollution coming from paper.

9. Conclusion

This project proved to be a great challenge, but more importantly a learning experience. I started the project with almost no experience in Web or mobile development and after months of struggling with the concepts and structures of Web and mobile development and especially of the technologies used. I am surprised by how much I have improved. Tasks that used to take me a day now take me less than an hour.

In addition to the technical skills acquired during this period, I have acquired skills that are applicable outside of the field of web or mobile development. These include better research and learning skills, improved debugging skills and intuition, as well as a deeper appreciation for developing solutions that will be deployed and that therefore need to be guarded against attacks and made secure, stable, and accessible.

10. Future Work

10.1. Handling Donations

The promotional website still does not have the functionality to handle donations and send them to an account of the NGO. This is because in order to establish a trusted and secure
channel and in order to link it with an account for the NGO, I would have to be present in Biougra (their physical location) in order to have full presence of the director of the NGO and to consult him at every step of the process. This will be completed during the latter half of May and the first half of April.

10.2. Deployment

The project still needs to be deployed, we are already in contact with organizations that provide free deployment for NGOs. I will have to prove that I am working for an NGO and send the required documents in order to get free deployment privileges. Otherwise, we will deploy on a paid platform since the NGO already agreed to finance the deployment if it can’t be made for free.

10.3. Integrating Arabic

We still need to add Arabic to the language file and integrate it within the products. This should be an easy task since the products with built with the future integration of Arabic in mind.
11. References


