AUTOMATION OF STEP BY STEP PROCEDURES

Hamza Touhs

Supervised By

Dr.Chraibi Mhammed

©SCHOOL OF SCIENCE & ENGINEERING – AL AKHAWAYN UNIVERSITY
ACKNOWLEDGEMENTS

I consider this capstone experience to be a very unique and interesting opportunity. Thanks to Dr. Mhammed Chraibi and Al Akhawayn University, I had the honor and privilege to explore the fascinating field of software engineering, apply the skills that I have learned during my journey at AUI, and most importantly, broaden my knowledge in software development and design.

Furthermore, I would like to express my sincere gratitude to both my supervisor Dr. Chraibi Mhammed, and my advisor Dr. Assem Nasser who managed to guide me throughout my capstone project as well as my academic experience as a whole, providing continuous advice and precious help. Owing to their generosity and vineyard I was able to develop my passion towards computer science.

Finally, exceptional recognition goes to my family and friends for their generous emotional support. They showed a considerable support, understanding, and patience throughout my experience with this capstone project.
Table of Contents

ACKNOWLEDGEMENTS ........................................................................................................... 3
ABSTRACT ................................................................................................................................. 8

1. INTRODUCTION .................................................................................................................. 9

2. STEEPLE Analysis ............................................................................................................... 10
   2.1 Social ............................................................................................................................. 10
   2.2 Technologically ............................................................................................................. 10
   2.3 Economical ................................................................................................................... 11
   2.4 Environmental ............................................................................................................. 11
   2.5 Political ......................................................................................................................... 11
   2.6 Legal ............................................................................................................................. 12
   2.7 Ethical .......................................................................................................................... 12

3. Feasibility: ........................................................................................................................... 13
   3.1 Financial Feasibility: ..................................................................................................... 13
   3.2 Technical Feasibility: .................................................................................................... 13
   3.3 Resource Feasibility: .................................................................................................... 13
   3.4 Risks: ............................................................................................................................ 14

4. SOFTWARE SPECIFICATION: .......................................................................................... 15
   4.1 Dashboard’s Software Specification: .......................................................................... 15
      4.1.1 Dashboard’s Functional Requirements: ............................................................... 15
      4.1.2 Dashboard’s Non-Functional Requirements: ....................................................... 16
      4.1.3 Dashboard’s Use Cases: ..................................................................................... 18
      4.1.4 Dashboard’s Sequence Diagram: ......................................................................... 18
   4.2 Mobile Application’s Software Specification: ............................................................. 21
      4.2.1 Mobile Application’s Functional Requirements: ................................................ 21
      4.2.2 Mobile Application’s Non-Functional Requirements: ....................................... 23
      4.2.3 Mobile Application’s Use Cases: ....................................................................... 23
      4.2.4 Mobile Application’s Sequence Diagram: ......................................................... 23

5. SOFTWARE ARCHITECTURE ............................................................................................. 24

6. Design ................................................................................................................................ 25
   6.1 Prerequisites .................................................................................................................. 25
6.2 Installing JS/UML.............................................................................................................25
6.3 Class Diagram ...............................................................................................................26

7. IMPLEMENTATION DETAILS OF THE SOLUTION..................................................26
7.1 Storing data in the database .......................................................................................27
7.2 Modify data from the database .................................................................................27
7.3 Completing a step ......................................................................................................27
7.4 Expiration of an event ..............................................................................................29
7.5 Mailer .........................................................................................................................29
7.6 Middlewares ...............................................................................................................30
7.8 Generating Qr Code ..................................................................................................31

8. TECHNOLOGY ENABLERS: .......................................................................................33
8.1 react-native ...............................................................................................................33
8.2 Postman: ....................................................................................................................33
8.3 MongoDB:....................................................................................................................34
8.4 Robo 3T: ......................................................................................................................34
8.5 Node.js .......................................................................................................................34
8.6 Git ...............................................................................................................................35
8.7 Bitbucket ....................................................................................................................35
8.8 GitBook .......................................................................................................................36
8.9 GitKraken....................................................................................................................36
8.10 Sources used for learning: .....................................................................................37

9. Future work ................................................................................................................38
10. Conclusion ................................................................................................................39
11. REFERENCES............................................................................................................40
List of Figures:

Figure 1: Dashboard’s Use Cases .................................................................................. 18
Figure 2: Dashboard’s Sequence Diagram for a Get Request ........................................ 19
Figure 3: Dashboard’s Sequence Diagram for a Post Request ......................................... 20
Figure 4: Dashboard’s Sequence Diagram for a Cron Job ............................................... 20
Figure 5: Dashboard’s Sequence Diagram for Failed Request ......................................... 21
Figure 6: Mobile Application’s Use Case .......................................................................... 23
Figure 7: Backend’s Class Diagram .................................................................................. 26
Figure 8: Postman’s Interface .......................................................................................... 33
Figure 9: Robo 3T’s Interface ......................................................................................... 34
Figure 10: Bitbucket’s Interface ....................................................................................... 36
Figure 11: GitKraken’s Interface ...................................................................................... 37
List of Tables:

Table 1: Functional Requirements of the Dashboard .................................................. 15
Table 2: Non-Functional Requirements of the Dashboard .......................................... 17
Table 3: Mobile Application’s Functional Requirements ............................................. 23
ABSTRACT

Several issues have been raised at Al Akhawayn due to miscommunications or misunderstandings of the intended steps of several procedures/forms. This project is an attempt to reduce the frustration and pain a student/faculty/staff can endure during the execution of a step by step procedure. The solution presented in this project allows its users to easily and efficiently look into at the exact steps of every procedure at Al Akhawayn University. Moreover, event administrators can have access to valuable data; such as, tracking the number of completions, the steps in which people struggle with, or uncompleted steps of each user. The solution presented in this paper is an online mobile application that uses state of the art technologies as well as follow the most recent standards and protocols. The software is compatible with both ANDROID and IOS and guaranties a non negligible security against malicious attacks.

Keyword: forms, procedure, data collection, mobile application, track, compatibility, security
1. INTRODUCTION

This capstone project consists on the development of a mobile application that automates step by step procedures within Al Akhawayn University. The developed software would aim on reducing the complexity of the paper work at the university. The idea was a suggestion from my current capstone supervisor, Dr. Chraibi Mhammed; and having personally went through the struggle of not having clear direction on what is expected from me, it was a great incentive and source of motivation to solve this issue for the upcoming generation of student at Al Akhawayn University. According to Dr. Stephen Stanfield, and Mrs. Bridget Candy, a confusing paper work and intense bureaucracy are prospective risk factor for common mental disorders (Stansfeld & Candy, 2006). This project is an attempt to alleviate such hardships on both students, and faculty/staff by facilitating following procedures, keeping track of the personal progression of each member, as well initiating and introducing new events to the community.

We intend to achieve this objective by implementing a mobile application that would be supported by IOS and ANDROID based devices. This application will present different functionalities for different users:

1. Administrators: They will be able to create “events” and manage the requirements needed to fulfil as well as the people enrolled in a created event. In addition, they will be able to scan user’s generated QrCodes to complete steps and assign other users as “scanners”.

2. Scanners: They are users that were given scanning privileges and are able to scan other users’ QrCode to complete a step.

3. Students/Parents: They will be able to join events and access a check list of the appropriate steps. To fulfil a requirement, the student/parent will need to find the scanner for that particular step. Typically, the scanner will be present in the location given by the application. As soon as it is scanned both the student/parent and the administrator will be notified of the accomplishment.

The minimum objectives would be a working backend in NodeJS that handles the creation of accounts, creation of events, set up of the requirements to fulfil, and provision of the appropriate QR Codes. Moreover, we will provide a frontend application supported by both IOS and ANDROID in React Native that equivalently supports the backend features. Both of the backend and frontend will follow the standard implementation techniques and security precaution as well as have a readable documentation.
2. STEEPLE Analysis

2.1 Social

This solution mainly targets the social aspect of the issue faced. This application will alleviate an important number of hassles that AUI’s community face during the completion of any event. The main problems that we are trying to solve are the follow:
Not knowing the full procedure to fulfil a requirement; hence, people rely on services which reply to the same answers over and over.
Having troubles keeping track of the completed steps and the remaining ones. This solution will provide its users with an automatic check list that dynamically presents the user valuable information.

Event/procedure administrators do not have access to any information regarding the completion of the steps. This application will be able to provide a detailed and tailored reports which can have several uses; such as, gather data to find trends and be able to present procedures with better design, or provide an accurate help to the user when in need.
Instead of keeping in mind all the steps to achieve a goal. Users will have to register to a particular procedure and solely focus on the step provided by the software.

2.2 Technologically

This project uses state of the art technology to provide the best functionalities and security level. The project is composed of four main part. The first one being the backend (also called “server-side”) of the application. It refers to the actual operations done in the server side which are all the features that the application presents. We decided to follow the REST software architecture. Besides the fact that it to most used protocol nowadays, RESTFUL APIs’ advantage lays in the fact that it totally separates the user interface from the server’s. This separation allows a language-constraint-free development. In fact, because building a Rest web API is supported by several programming languages. It allowed us to delay the development of the frontend in order to the decide on the best technology to use. Because we intend on developing a mobile application that would be compatible for both ANDROID and IOS, we decided on developing using React Native, one of the most popular JavaScript libraries for frontend development. This powerful technology allows the development on both platforms using one source code. The third deliverable is a dashboard from which event/procedure administrators can manage their events as well as the users. Only trusted member with admin privileges will have have access to this page. The fourth
deliverable is a (.md) file that represents the documentation of the application. It will contain guidelines to use the software as well as implementation details so facilitate the customization and the maintenance phase of the system.

Concerning the other tools and software:

- **Postman**: it is an API testing software that allow its user to simulate API calls without having to develop a front end design. Currently the handled requests are: GET, POST, PUT, PATCH, DELETE, COPY, HEAD, OPTIONS, LINK, UNLINK, PURGE, LOCK, UNLOCK, PROPFIND, and VIEW. The software sends the corresponding request either to an online or local server and displays the response in a JSON format (default: Restful API calls). I was using this software throughout the project in order to test the backend program.

- **MongoDB**: it is the most widely used open-source cross-platform NoSQL database. It uses a JSON format to store and output the data using schemata. We decided to use this technology instead of the commonly used SQL/MYSQL database management system because of its ease of use and its useful feature; such as: indexing, aggregation, and server-side JavaScript execution [3]. MongoDB does not have an official interface, rather it is used using the command line/.bash which make its use slightly more complex.

### 2.3 Economical

This software doesn’t seem to have an impact on the economy as it is indented to be a free mobile application that has as soul purpose serving AUI’s community; therefore, the application is vulnerable to inflation and fluctuations.

### 2.4 Environmental

At first glance the application does not have any repercussions or impact on the environment. We intend on doing further research after the development of the software. The research will consist on calculating the energy cost of the application of an average user, estimate the number of users and figure out if this solution is worth from an environmental perspective.

### 2.5 Political

Besides facilitating bureaucratic procedures which, in theory, will be beneficial for the community, this application does not seem to have an impact on the political ground.
2.6 Legal

The software will abide by the rules of Al Akhawayn University. A “terms & conditions” needs to be agreed upon by every user before proceeding to using the application. Failing to comply with Al Akhawayn’s regulation can result in fines, or legal actions; hence, it is primordial to notify the university for any important design change. Additionally, we will be exclusively using open source frameworks, libraries, and npm packages.

2.7 Ethical

From an ethical perspective, we know that our application can be used for malicious intent and can lure some students; therefore, we will have to find fully secure the application; moreover, I feel that it is my ethical duty as a student and developer to push for as much student agency and freedom in the app; though, I do not have the final say on what features exist in the app, I am in a position where I can make strong suggested backed by my strong experience as an AUI students.
3. Feasibility:

3.1 Financial Feasibility:

Because this project is an online application software, there will be hosting costs associated to it. Al Akhawayn University’s data centers will be needed. Because the application won’t be used to transfer large data files, a high bandwidth is not needed to sustain the application or make it usable.

3.2 Technical Feasibility:

We decide to choose state of the art technology as it provides a very active community that updates and manages the optimization details as well as the security details of those tools. The main languages and software that will be used is:

- HTML: Shall be used to generate the content of the dashboard on a browser.
- CSS: Shall be used to tune the style and layout of the dashboard.
- JavaScript: Shall be used to make API requests for the database.
- NodeJS: Shall be used to develop the backend; i.e: develop the necessary APIs routes.
- React-Native: Shall be used to develop the frontend for both Android and IOS.
- MongoDB: Shall be used as the database management system of this application.
- Git: Shall be used for version control.
- Postman: Shall be used in order to test the backend API route without having to develop a graphical user interface during the development phase.

All of those tools are freely available online. React-native is the only tool that was new to me and to which I needed to dedicate some time to learn. All of those tools are well known for being very well designed and reliable. Initially, we will be using free web mailers and hosting services in order to test the application while it is in a development stage; however, it will be hosted on AUI’s server when deployed.

3.3 Resource Feasibility:

One of the main advantages of computer science based project is that it rarely needs a complex or expensive resources. In this case we need a programming device, my personal computer and the university’s lab computers will be used. A hosting space; free options are readily available online; and programming tools, which are available for free.
3.4 Risks:

The first and most important risk that we face developing this software is the limited time that we dispose of in order to learn new technologies, and maintain an acceptable academic performance in other classes. Another risk that we might face as software developers, is the fragility of the system. We plan on solving this issue by apply cutting edge security frameworks and follow the security norms and standards.
# 4. SOFTWARE SPECIFICATION:

## 4.1 Dashboard’s Software Specification:

### 4.1.1 Dashboard’s Functional Requirements:

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Requirement Statement</th>
<th>Must/Want</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR101</td>
<td>The dashboard shall allow the administrators only to log in</td>
<td>Must</td>
<td>Only the administrators will have access to the dashboard software</td>
</tr>
<tr>
<td>FR102</td>
<td>The dashboard shall allow the admin to retrieve the password of their accounts</td>
<td>Must</td>
<td>The software should prompt the user to enter the e-mail of their account in which they will receive a code. This code will be used to change the password in case it was forgotten</td>
</tr>
<tr>
<td>FR103</td>
<td>The dashboard shall allow the users to create new accounts</td>
<td>Want</td>
<td>After creating the account from the website, the user won’t be able to connect as the dashboard only handles administrator accounts</td>
</tr>
<tr>
<td>FR104</td>
<td>The dashboard shall present the administrator with statics in the home page</td>
<td>Want</td>
<td>The statics that are going to be displayed are the number of users, the number of completions, the number of users still following the</td>
</tr>
<tr>
<td>FR105</td>
<td>The dashboard shall allow the administrator to view all users</td>
<td>Must</td>
<td>Administrators will have access to all the registered users</td>
</tr>
<tr>
<td>FR106</td>
<td>The dashboard shall allow the administrator to add new users</td>
<td>Must</td>
<td>-</td>
</tr>
<tr>
<td>FR107</td>
<td>The dashboard shall allow the administrators to view all the events as well as their steps</td>
<td>Must</td>
<td>Administrators should be able to create events (only administrators are allowed to create events) and view the respective steps of each event</td>
</tr>
<tr>
<td>FR108</td>
<td>The dashboard shall allow the user to view the other administrator accounts</td>
<td>Must</td>
<td>-</td>
</tr>
<tr>
<td>FR109</td>
<td>The dashboard shall allow the administrators to give “scanner right/privileges” to user</td>
<td>Must</td>
<td>An administrator should have the possibility to assign a user as a scanner for a particular step</td>
</tr>
<tr>
<td>FR110</td>
<td>The dashboard shall allow the user to sign out of the platform</td>
<td>Must</td>
<td>-</td>
</tr>
</tbody>
</table>

*Table 1: Functional Requirements of the Dashboard*
4.1.2 Dashboard’s Non-Functional Requirements:

The non-functional requirements of the dashboard and mobile application are the same as both software uses the same API Calls. The only different is the frontend, i.e: the way and medium in which data is displayed.

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>User Requirement Statement</th>
<th>System Requirement</th>
<th>Must/Want</th>
</tr>
</thead>
</table>
| NFR101         | The system should be user friendly and easy to navigate | • The program shall provide an easy, intuitive interface.  
                  |                             | • The interface shall be using meaningful icons and/or descriptive words  
                  |                             | • The system’s features should be easy to master | Must |
| NFR102         | The system should not take a large amount of memory space | • Program shall not exceed more than 5 Megabytes during execution  
                  |                             | • The system shall use optimized data structures | Want |
| NFR103         | The system should have a fast run-time | • System shall be able to complete tasks without a noticeable lag in runtime.  
                  |                             | • The system should not take more than 5 seconds to open  
                  |                             | • Performed actions should not take more than 2 seconds | Want |
| NFR104         | Users should not crash unexpectedly | • The system should not have a failure rate of more than 2%.  
                  |                             | • User data shall be protected if a crash occurs | Must |
| NFR105         | Dashboard: The user shall run on the most used browsers and operating systems  
                  | Mobile Application: The user shall run on IOS and Android operating systems | Dashboard: The user shall run on the most used browsers and operating systems  
                  |                             | Mobile Application: The user shall run on IOS and Android operating systems | Must |
| NFR106         | The system shall be delivered by May 10, 2019 | • The software shall be fully designed, developed, and tested by May 10, 2019 | Must |
| NFR107         | The system development shall follow the IEEE | The system development shall follow the IEEE Software and | Must |
| NFR108 | The system shall interoperate correctly with computer files and hardware | • The system shall be compatible with computer hardware  
• The system should read correctly from computer files | Must |
| NFR109 | The users shall not use the system for unethical business | • The system should not be commercialized to non-trusted businesses and individuals.  
• Software files of an organization shall not be communicated to another organization unless it is with managers’ agreement and for ethical purposes. | Must |
| NFR110 | The user shall not be able to access any personal information but his or her own, with the exception of the manager who may access all information on the system | The system shall not display any personal information of any user who interacts with the system at any point to any user if that user does not have granted privileges to view it | Must |
| NFR111 | The system should not harm computer resources and user data | • The system should not overuse battery resources.  
• The system should be viruses and malware free. | Must |
| NFR112 | The system shall protect data generated by software users. | The system shall use encryption in order to secure data and information from being stolen. | Must |

Table 2: Non-Functional Requirements of the Dashboard
4.1.3 Dashboard’s Use Cases:

**Procedure Automation Dashboard**

![Diagram of Procedure Automation Dashboard](image)

Figure 1: Dashboard’s Use Cases

4.1.4 Dashboard’s Sequence Diagram:

4.1.4.1 Dashboard’s Success Scenario:

Similarly to the functional requirements, both the dashboard and the mobile application uses the same API call, this means that they share similar use cases. The following use cases summarize the main types of requests the software make use of:
• **Get Requests:**

  Get requests are requests to the Nodejs backend that expect data to be sent back. Because we are following software engineering standards, i.e: REST, the data is sent back in a JSON format.

![Dashboard's Sequence Diagram for a Get Request](image)

*Figure 2: Dashboard’s Sequence Diagram for a Get Request*
• **Post Requests:**

Post requests are requests that are used when the user needs to alter the data stored in the database. To increase the security, convenience, and portability, restful APIs are used.

![Figure 3: Dashboard’s Sequence Diagram for a Post Request](image)

• **Cron Jobs:**

Cron jobs are a time-based software utility that get triggered in a previously scheduled time and date. An example of such functions would be the notification sent to the user to notify them that an event has expired. In this case, the user is not supposed to make any request, whenever an event has expired, node.js will update the status of the concerned event in the database and notify the user of the change.

![Figure 4: Dashboard’s Sequence Diagram for a Cron Job](image)
4.1.4.2: Dashboard’s Failure Scenario:

Figure 5: Dashboard’s Sequence Diagram for Failed Request

4.2 Mobile Application’s Software Specification:

4.2.1 Mobile Application’s Functional Requirements:

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Requirement Statement</th>
<th>Must/Want</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR201</td>
<td>The mobile application shall allow the user to create an account</td>
<td>Must</td>
<td>Users will be able to register into the database</td>
</tr>
<tr>
<td>FR202</td>
<td>The mobile application shall allow the user to modify their passwords</td>
<td>Must</td>
<td>The mobile application should prompt the user to enter the e-mail of their account in which they will receive a code. This code will be used to change the password</td>
</tr>
<tr>
<td>FR203</td>
<td>The mobile application shall allow the user to login by submitting either their email or their AUI ID</td>
<td>Must</td>
<td>All the users should be able to login to the mobile application and get directed to the home page</td>
</tr>
<tr>
<td>FR204</td>
<td>The mobile application shall allow the users to assess their progression</td>
<td>Must</td>
<td>Users will be able to assess the progression of each event that has been previously joined</td>
</tr>
<tr>
<td>FR205</td>
<td>The mobile application shall allow the user view the information about a step</td>
<td>Must</td>
<td>Users should be able to view all the steps of an event as well as the appropriate information about the event.</td>
</tr>
<tr>
<td>FR205</td>
<td>The mobile application shall allow the user to browse/search across all the events</td>
<td>Must</td>
<td>The user should be able to access all the event</td>
</tr>
<tr>
<td>FR207</td>
<td>The mobile application shall allow the user to filter the events and access only the ones the user has previously joined</td>
<td>Must</td>
<td>The user should be able to access the joined events</td>
</tr>
<tr>
<td>FR208</td>
<td>The mobile application shall allow the user to view their profile information as well as modify the basic ones</td>
<td>Must</td>
<td>The user should be able to view and modify, the first name, last name, email address, phone number, and country</td>
</tr>
<tr>
<td>FR209</td>
<td>The mobile application shall allow the user to view their Qr Code and display it in the screen to be scanned</td>
<td>Must</td>
<td>The application should be able to generate a unique Qr Code</td>
</tr>
<tr>
<td>FR210</td>
<td>The mobile application shall notify the user when an event is nearly expired</td>
<td>Want</td>
<td>The application should send a notification to the user when and event is expiring soon</td>
</tr>
<tr>
<td>FR211</td>
<td>The mobile application shall allow the users to view the steps that they can scan</td>
<td>Must</td>
<td>The users should be able to view the steps that can be scanned using their accounts. This scanning privilege can be given to a user by an administrator only</td>
</tr>
<tr>
<td>FR212</td>
<td>The mobile application shall allow the users to scan scan the Qr Codes of other users willing to complete a step</td>
<td>Must</td>
<td>The users can scan other user’s Qr Codes and mark them as completing a step</td>
</tr>
<tr>
<td>FR213</td>
<td>The mobile application shall allow the users to join and leave event</td>
<td>Must</td>
<td>-</td>
</tr>
<tr>
<td>FR214</td>
<td>The mobile application shall allow the user to view the notifications</td>
<td>Want</td>
<td>Users should be able to view the notification received</td>
</tr>
<tr>
<td>FR215</td>
<td>The mobile application shall allow the users to view the location in a map of a step</td>
<td>Want</td>
<td>Users should be able to view the location of to achieve the next step</td>
</tr>
<tr>
<td>FR216</td>
<td>The mobile application shall allow the user to</td>
<td>Want</td>
<td>Users should be able to scan a QR code to complete a step</td>
</tr>
</tbody>
</table>
Table 3: Mobile Application’s Functional Requirements

| FR217 | The mobile application shall allow the user to Logout | Must | - |

4.2.2 Mobile Application’s Non-Functional Requirements:

The mobile application and the dashboard share similar non-functional requirements. Please refer to the section above (7.1.2) to find the non-functional requirements.

4.2.3 Mobile Application’s Use Cases:

![Diagram of Mobile Application’s Use Case]

Figure 6: Mobile Application’s Use Case

4.2.4 Mobile Application’s Sequence Diagram:

Both software are using the same node.js API calls. Please refer to the section above (7.1.3) to find the sequence diagram.
5. SOFTWARE ARCHITECTURE

This software uses the restful software architecture. The database is accessed only by the backend server and the frontend is never allowed to alter or establish a connection with the database. For the software to be called restful, it need to abide by 6 principles [2]:

1. Client-Server: This means that a separation needs to be established between the user interface and the data storage. This improves the portability across multiple platforms as well as the scalability of the project.

2. Stateless: This refers to the fact that the request must contain all the information to get the result back and is not allowed to make use of any storage; in other words, one request per response.

3. Cacheable: Restful architectural style states that whether a response is allowed to be cached or not is necessary. When sending any data, stating its “cacheability” is required.

4. Uniform Interface: According to Rest documentation, restful APIs is defined by four interface constraints: “identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state” [2].

5. Layered System: This principle refers to encapsulating data so that it is not accessed by outsider components. This architecture is composed of hierarchical structures that prevent components from accessing layers they are not supposed to interact with.

6. Code on Demand: It allows running code as applets/scripts which simplifies the coding phase as an important number of features will be re-implemented.
6. Design

For a more accurate and much faster generation of the class diagram, we have opted to use an eclipse plug-in that creates a UML diagram from a Javascript code. Because this tool is extremely useful and the documentation does not seem to include all the steps to make use of it. We have decided to include a small tutorial to explain the installation procedure as well as a quick start.

6.1 Prerequisites

- Download file JS/UML 0.9.3^ (or above) (https://sourceforge.net/projects/jsuml/files/)
- Install Eclipse IDE v4.5.2^ (or above)
- Install “Papyrus UML” 8.1.1 Installation of Papyrus UML
  - Select Help > Install New Software menu item
  - Add new site by press the “Add…” button
  - Input the following data: Name: Papyrus Mars updates | Location: http://download.eclipse.org/modeling/mdt/papyrus/updates/releases/mars
  - Selecting “Papyrus” from the list view > Next
  - Accepting the terms of the license agreement > Finish

6.2 Installing JS/UML

- Help > Install New Software
- Click on the “Add…” button > “Archive…”
- Choose the Js/UML.zip file previously downloaded
- Follow the installation wizard
- Restart Eclipse
6.3 Class Diagram

![Class Diagram Image]

Figure 7: Backend’s Class Diagram
7. IMPLEMENTATION DETAILS OF THE SOLUTION

7.1 Storing data in the database

To store and retrieve data from the database, we had to make use of a library called “Mongoose”. This library provides a straight-forward solution to model and update entry documents in mongodb using Node.js. Here is a sample code:

```javascript
static createNewStep(eventID, stepTitle, stepDescription) {
    let newStep;
    let hashedID = "STEP" + Hasher.generate();
    newStep = {
        eventID,
        stepTitle,
        stepDescription,
        hashedIDStep: hashedID
    };
    // Create new event in database
    return this.create(newStep)
}
```

In this example, we are creating a new step entry. First of all, the function gets as parameters the eventID which is the ID of the event the step belongs to, the step’s title and its description. We then create an object that we call newStep as well as a variable called hashedID. In this latter, we will generate a random string and append “Step” to it. The purpose of this hashedID is to protect the mongodb generated _id as we need a unique key that will refer to this object and be available publically. Finally we store the data in the previously instantiated object called newStep and call the function create() that is defined in mongoose library and pass our object as parameter. After the execution of this line, a document will be created in MongoDB.

7.2 Modify data from the database

In order to modify an entry, we also have to make use of function from mongoose. Several alternative functions are available, one of the most frequently used ones is findOneAnUpdate({attribute},{update}). This function takes as argument an attribute of the element we want to update, as well as the modification that we want to apply.

7.3 Completing a step

To handle this requirement, the following route has been created:

```javascript
/**
 * Complete a step (needs to be either the scanner or the admin of the event)
 * @type {Router|router}
 */
```
router.get('/hashedIDScanner/events/:eventHashedID/steps/:stepHashedID/user/:hashedIDUser/complete', appAuthVerify, (req, res) => {
  let hashedIDScanner = req.params.hashedIDScanner;
  let eventHashedID = req.params.eventHashedID;
  let stepHashedID = req.params.stepHashedID;
  let hashedIDUser = req.params.hashedIDUser;

  let thisUser;
  let thisEvent;
  let thisOtherUser;

  User.findByHashedID(hashedIDScanner)
    .then(user => {
      if (!user)
        throw new ServiceError(ERROR_VALUE.account_does_not_exist);
      thisUser = user;
      return User.findByHashedID(hashedIDUser)
    })
    .then(user => {
      if (!user)
        throw new ServiceError(ERROR_VALUE.account_does_not_exist);
      thisOtherUser = user;
      return Event.findByHashedID(eventHashedID)
    })
    .then(event => {
      if (!event)
        throw new ServiceError(ERROR_VALUE.event_not_exist);
      if (!thisUser.scannerFor.includes(stepHashedID) && event.adminID !== hashedIDScanner)
        throw new ServiceError(ERROR_VALUE.user_not_admin_nor_scanner);
      thisEvent = event;
      return Step.findByHashedID(stepHashedID)
    })
    .then(step => {
      if (!step)
        throw new ServiceError(ERROR_VALUE.step_not_exist);
      if (step.eventID !== eventHashedID)
        throw new ServiceError(ERROR_VALUE.step_not_belong_to_event);
      return Event.isUserParticipating(eventHashedID, hashedIDUser)
    })
    .then(isUserParticipating => {
      if (!isUserParticipating)
        throw new ServiceError(ERROR_VALUE.user_not_participating);
      return Step.isAlreadyCompleted(stepHashedID, hashedIDUser)
    })
    .then(isAlreadyCompleted => {
      if (isAlreadyCompleted)
        throw new ServiceError(ERROR_VALUE.step_already_completed);
      return Step.complete(stepHashedID, hashedIDUser)
    })
    .then(data => {
      return Step.checkIfFullyCompleted(thisEvent, thisOtherUser)
    })
    .then(async isFullyCompleted => {
      //add user to the list of fully completed
      if (isFullyCompleted)
        thisEvent.fullyCompletedIDs.push(thisOtherUser.hashedID);
      await thisEvent.save();
    });
First of all, we have to perform the following checks:

- Is the token correct and still valid?
- Does the scanner-user exist?
- Does the scanned-user exist?
- Does the event exist?
- Does the step exist?
- Does the step belong to the event found?
- Does the scanner-user has enough privilege to execute such operation?
- Is the scanned-user participating to the event?
- Has the step already been completed by the scanned-user?

In case of the checks was invalid, the server sends the appropriate error message; otherwise, the scanned-user completes the step by adding his/her ID to an array containing the IDs of people who have completed the step.

### 7.4 Expiration of an event

To perform this operation, we have to make use of cron jobs. The server runs every day at 12:00am and performs a check on all the non-expired events. In case an event has expired (expirationDate < Date.now ), the state of the event is modified to “expired” and and notification is sent to all the people who have joined that event.

### 7.5 Mailer

During the development phase, we were using a mailing service named MailGun. This platform offers free mailing services; however, the free version mainly serves as a testing mailer because the limit number of mails is very limited for a real application context [2]. When deploying the application, we are planning on using AUI’s mailing servers.
When creating an account with a mailing server, we usually expect an API-Key and a domain. That information will be used for the authentication. Then we simply have to specify some options:

1. let activationMailOptions = {
2.   from: "Capstone <no-reply@capstone.io>",
3.   to: "",
4.   subject: "Capstone- Activate Account",
5.   template: "activation/activation",
6.   context: {
7.     recoveryUrl: ""
8.   }
9. };
10.

Note: the template contains an .hbs (activation.hbs) file specifying the format of the mail and its design.

7.6 Middlewares

Middlewares are functions that have access to the request received and response sent by the server. A middleware can execute any code, modify the request and response, end the server’s cycle, and call another middleware \[^6\].

Let’s see as an example, the middleware used in the previous example (signUpVerify):

1. "use strict";
2. 
3. const {ServiceError} = require("../service/service-error");
4. 
5. // Checks the fields of the JSON sign up request
6. let signUpVerify = (req, res, next) => {
7.   req.checkBody("displayName")
8.     .exists().withMessage("displayName_required")
9.     .isString().withMessage("displayName_not_string")
10.    .isLength({min: 1, max: 30}).withMessage("displayName_exceed_length")
11.    .matches(/^[a-zA-Z0-9\s._-]*$/).withMessage("displayName_invalid_char");
12. 
13.   req.checkBody("email")
14.     .exists().withMessage("email_required")
15.     .isString().withMessage("email_not_string")
16.     .notEmpty().withMessage("email_required")
17.     .isEmail().withMessage("email_invalid");
18. 
19.   req.checkBody("phoneNumber")
20.     .exists().withMessage("phoneNumber_confirm_required")
21.    .isString().withMessage("phoneNumber_confirm_required_not_string")
22.    .matches(/^[0-9]*$/).withMessage("phoneNumber_invalid_char");
23. 
24.   req.checkBody("password")
25.     .exists().withMessage("password_required")
26. };
27. 
28. // End of middleware
This middleware checks if the request contains a `displayName`, `email`, `phoneNumber`, `password`, and `confirmPassword`; while making sure that each field is in the expected format. In case the request is not as expected, the code within the route won’t execute, instead a response containing an explanation in restful response (JSON).

Note that this middleware does not check if the `username` has been sent or not. This is because one of the requirements set by the client, is that the `username` must be optional. Therefore, if the user doesn’t enter a `username` we automatically set a generated hashed value.

### 7.8 Generating Qr Code

To generate the Qr Code of each user, we make use of a react-native library called “react-native-qrcode”. The library provide a plug and play straightforward way to generate Qr Codes.

```jsx
<QRCode
  value={this.state.hashedID}
  bgColor="#201747"
  squared />
```
The following piece of code, creates a squared QrCode and uses “#201747” as main color to draw the code. Concerning the value attribute, it converts the hashedID of the user stored in the state of the page. This is an example that displays the use of creating a hashed version of the “_id” provided by mongoDB. Due to safety reason, the “_id” should remain private and should never be sent to the end user.
8. TECHNOLOGY ENABLERS:

8.1 react-native

React native is a framework developed by Facebook. It is a revolutionary technology as it solved important issues the app development industry was suffering from. When developing a mobile application, developers had to develop two different applications. One of them designated for IOS platform and another designed for Android. IOS runs C# and Swift natively, while Android runs Java and Kotlin. React native approach is pretty smart as it does not create a jvm-like mechanism that runs on top of the devices to all the javascript code to run on both machines, instead, react native converts javascript code into native languages. In other words, it converts javascript into C# and Java. This results in a fast and cheap development phase; moreover, a great performance as the resulted software is a native code. [4]

8.2 Postman:

Postman is an API testing software that allow its user to simulate API calls without having to develop a front end design. Currently the handled requests are: GET, POST, PUT, PATCH, DELETE, COPY, HEAD, OPTIONS, LINK, UNLINK, PURGE, LOCK, UNLOCK, PROPFIND, and VIEW [5].

The software sends the corresponding request either to an online or local server and displays the response in a JSON format (default: Restful API calls).

I was using this software throughout the project in order to test the backend program.

Figure 8: Postman’s Interface
8.3 MongoDB:

MongoDB is the most widely used open-source cross-platform NoSQL database. It uses a format that is similar to JSON to store and output the data using schemata. We decided to use this technology instead of the commonly used SQL/MYSQL database management system because of its ease of use and its useful feature; such as: indexing, aggregation, and server-side JavaScript execution[6].

MongoDB does not have an official interface, rather it is used using the command line/.bash which makes its use slightly more complex.

The most important command is starting the database server which is done by going where MongoDB is installed then to /mongo/bin where you can execute the following command[6]:

```
```

8.4 Robo 3T:

Robo 3T is a free lightweight graphical user interface for MongoDB. It is a very simple yet really useful software that makes MongoDB even more simple to use[7].

In order to make the development faster, I was using Robo 3T to quickly and accurately view, delete, or alter documents. The discovery of this software was a game changer for me as I find it challenging to fully understand and view long documents in the terminal because of the limited size of the console.

Figure 9: Robo 3T’s Interface
8.5 Node.js

JavaScript was primarily used for client side scripting; such as, animations; however, this language starts getting more and more used in many other ways. Node.js is an open-source cross-platform JavaScript environment. This library allows its users to run JavaScript not in a browser but in server. This technology has made backend programming much more easy and intuitive. It’s main advantage over other alternative programming languages such as PHP, is its capacity to handle requests asynchronously. This design pattern ensures a non-blocking code execution \[^{[8]}\]. It was a bit confusing as generally speaking we tend to think of our code to flow line by line; however, with asynchronous programming, a piece of code might run before another even though the latter is written before the former.

8.6 Git

Git is the main version-control system. It is used to manage changes in a shared file among multiple people. Even though technically it can be very useful in a lot fields, it is mainly used by software developers \[^{[9]}\]. It is worth noting that it has been create by the famous Linus Torvalds in 2005; moreover, he has created this version-control system in order to manage Linux’s development. During my internship, we didn’t use git directly but went through other application that use git’s APIs; such as: Bitbucket or Github.

8.7 Bitbucket

Bitbucket is a web application that allows version control over repositories that get hosted in the cloud. It is similar to GitHub; both of them primarily use Git. Originality, this service was marketed to professional developers that want to keep a private source code \[^{[10]}\].

We were using Bitbucket instead of other more famous platforms like Github because Bitbucket allows having private repositories whereas Github requires the code to be open-source \[^{[10]}\]. It is true that Github offers a private repositories services; however, it is a paid feature that Bitbucket offers for free.
8.8 GitBook

Gitbook is a platform for hosting software’s documentations. It creates a “.md” file which is the main extension used today for writing documentations thanks to the markdown language \[11\]. We used this platform to write and store all the documentations (backend, android’s frontend, IOS’s frontend, and dashboard).

8.9 GitKraken

GitKraken is a cross-platform Git client; in other words, it is a software that links a very organized graphical user interface to the Git commands \[12\].

Personally, it took me a while before being able to grasp git’s methodology, and using the terminal to do sophisticated commands was quite challenging for me. Having a merge issue or branching was a nightmare; however, GitKraken made things much easier. Instead of thinking about the command themselves, I could focus more on more complex solution that would make a smoother development process.
8.10 Sources used for learning:

I started my learning process with a node.js course Udemy[^13]. It was very detailed and could grasp the main concepts of both JavaScript and Node; furthermore, YouTube channels such as “CodingTrain” were of a tremendous help[^14]. It was extremely easy to follow and the pace was fast enough to feel the evolution that motivated me to keep up. During the development process, StackOver Flow was clearly the main resource I look into whenever I’m facing a major problem or have any concern[^15].
9. Future work

In the course of my project, I intended on adding a research component that would tackle mobile application issues from other perspectives. One of the main research areas I want to focus on is improve the Moroccan user experience; in order words, find the optimal way to design an appealing mobile application for a Moroccan user. In fact, Moroccans have a distinct approach with regards to language; it is one of the few countries where the majority of its citizens are multilingual (El Mortaji, 2002). This adds quite an important challenge for software designers to choose the correct language and layout that would help the user locate more easily the information they are seeking. I will have to find out the language that would appeal to both parents and students as well as find a non-misleading design where users can navigate in without any difficulty.

Another interesting research that is can be beneficial for this mobile application has to do with topography. One of the features that we intend on implementing in our system is a guiding functionality; in other words, users will be able to be directed towards the next step they need to accomplish by showing they the location in a map; however, the major map provider (Google Maps) does not handle Al Akhawayn’s campus paths; therefore, we will have to design a map that would reflect our beautiful campus. This map can be used to provide a more accurate routing system. On a side note, we will have to design a short path finding system that would give the users the ability to move from a point to another in the most optimized way.

From an ethical perspective, we know that our application can be used for malicious intent and can lure some students; therefore, we will have to find fully secure the application; moreover, I feel that it is my ethical duty as a student and developer to push for as much student agency and freedom in the app; though, I do not have the final say on what features exist in the app, I am in a position where I can make strong suggested backed by my strong experience as an AUI students.
10. Conclusion

This application can be very helpful for AUI community as students/parents will be able to navigate more easily in AUI’s campus and will allow them to have a clear vision on the requirements needed to accomplish procedures. Moreover, this application can give a modern and “smart” reputation to Al Akhawayn University. On a more global level, this application can facilitate all the procedural tasks and be sort of an assistant that helps managing paperwork complexities.

On a different note, this experience allowed me to acquire very valuable skills that increase my value within the software development market. In fact, all the technologies that have been used are considered as being the next-gen tools and programming languages.

Finally, I would like to thank my friends, family, and my supervisor Dr. Chraibi Mhammed, who provided me with the support and technical knowledge that allowed me to learn the most out this capstone project.
11. REFERENCES


[16] El Mortaji, L. (2002). Writing ability and strategies in two discourse types: a cognitive study of multilingual Moroccan university students writing in Arabic (L1) and English (L3).
APPENDIX A

Screenshots of the dashboard:

Sign in:

Statistics Page:
APPENDIX B

Screenshots of the mobile application:

Sign in Page:

Sign up Page: