# 5. Implementation, Integration and Test Plan

## 5.1. Overview

It is possible to show the existence of bugs through program testing, but not to show their absence. In order to accomplish this goal, the aim is to find as many bugs as possible before the release date. All the preliminary considerations needed to implement and test eMSP and CPMS are presented and explained in this section.

## 5.2. Implementation plan

To implement the whole system, the bottom-up approach is selected so that each section of the system can be implemented and tested separately. The components and subcomponents described in the component view section can be divided.

in various subsystems:

* UserMobileApp
* CPOWebApp
* Application server
* External systems: DBMS, Google map service, Google calendar service.

The implementation should be done from the lower components up to the top ones. Because there are some components that rely on others. For example, every component and sub-system described in the component diagram interact with the DBMS. Thus, we implement the DBMS component. On the other hand, Google Maps and Google calendar are two other components that play a role like a database. So, they should be implemented in this step. After that, we can proceed to the implementation of the subcomponents in the Application Server component. Here, there are some main subcomponents such as the Model and the DataEncrypt subcomponents. Most components use them as a connection to access the database. Then, it is Mail and Authentication subcomponent’s turn. The Mail subcomponent should be implemented before the other one because the functionalities of Authentication rely on the Mail subcomponent, and we want to keep using the bottom-up approach while testing. In the next step, we can go for implementing all the subcomponents in the UserMobileApp and CPOWebApp. Each of these subcomponents works independently, so they can be implemented at the same level.

## 5.3. Integration and Testing

The following diagrams illustrate how the process of integration testing takes place, according to a bottom-up approach.

The arrows start from the component which “uses” the other one.

* At first the Model, DataEncrypt, and DBMS have been integrated and unit tested. They are not complex components, but other components rely on them to provide some services: that’s why they are integrated and tested first.

Diagram

Description automatically generated

Integration: Model, DataEncrypt, DBMS

* Then, as we mentioned in the previous section, two other important components named Authentication and Mail should be integrated with the Model component.

Diagram

Description automatically generated

Integration: Model, Mail, Authentication

* In the next step, the components responsible for gathering information about car status and maps have been integrated and tested for the suggestion components.

Diagram, engineering drawing

Description automatically generated

Integration: Suggestion, Geolocation, CarStatus

* The next two steps can be implemented and tested parallel. Because their functionalities are independent.

Diagram

Description automatically generated

Integration: Model, Geolocation, EditInfomartion,Pay