Cairo University  
Faculty of Computers and Information



**CS251**

**Software Engineering II**

BeHealthy

Software Design

2016/2017

Contents

[Instructions [To be removed] 3](#_Toc434318348)

[Team 3](#_Toc434318349)

[Document Purpose and Audience 3](#_Toc434318350)

[System Models 3](#_Toc434318351)

[I. System Decomposition 3](#_Toc434318352)

[II. Class diagrams 6](#_Toc434318353)

[Important Algorithm 7](#_Toc434318354)

[III. Sequence diagrams 8](#_Toc434318355)

[Class - Sequence Usage Table 9](#_Toc434318356)

[IV. Physical Entity-Relationship Diagram 10](#_Toc434318357)

[V. User Interface Design 10](#_Toc434318358)

[VI. Bonus: State Diagram and OCL 12](#_Toc434318359)

[Ownership Report 12](#_Toc434318360)

[Policy Regarding Plagiarism: 12](#_Toc434318361)

[References 12](#_Toc434318362)

[Authors 12](#_Toc434318363)

# Instructions [To be removed]

* **IMPORTANT. Rename this document to CS251-LeaderID-SDDocument.docx**

**(e.g. CS251-20040752-SDDDocument.docx)**

* **Remove the following notes and any red notes**
* **This document is the template document for your Software Design.**

# Team

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
|  | Donia Khaled |  |  |
|  | Hager Sobeah |  |  |
|  | Safwa Salman |  |  |
|  | Mahmoud Samy |  |  |

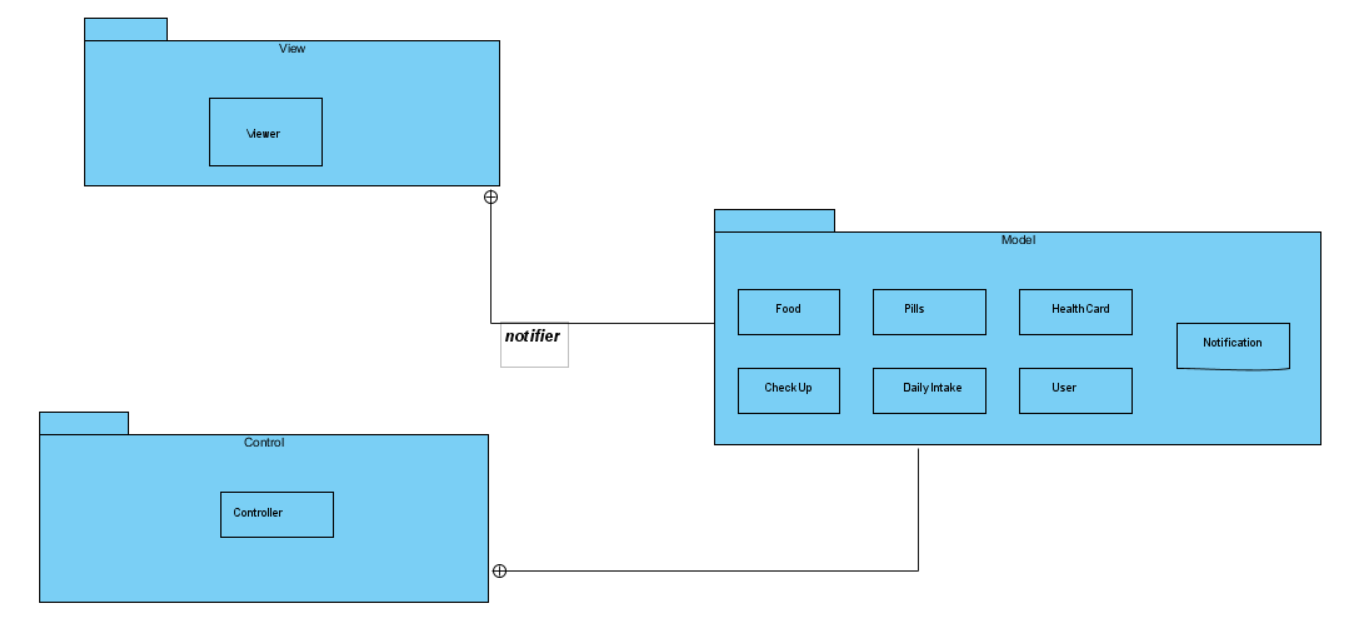
# Document Purpose and Audience

* **Any document should tell the reader 2 things: What is this document? Who is expected to read it?**
* **Write in simple notes: what this document is.**
* **List the target audience to read this document (e.g. CEO? Project Manager? Customer...?)**

# System Models

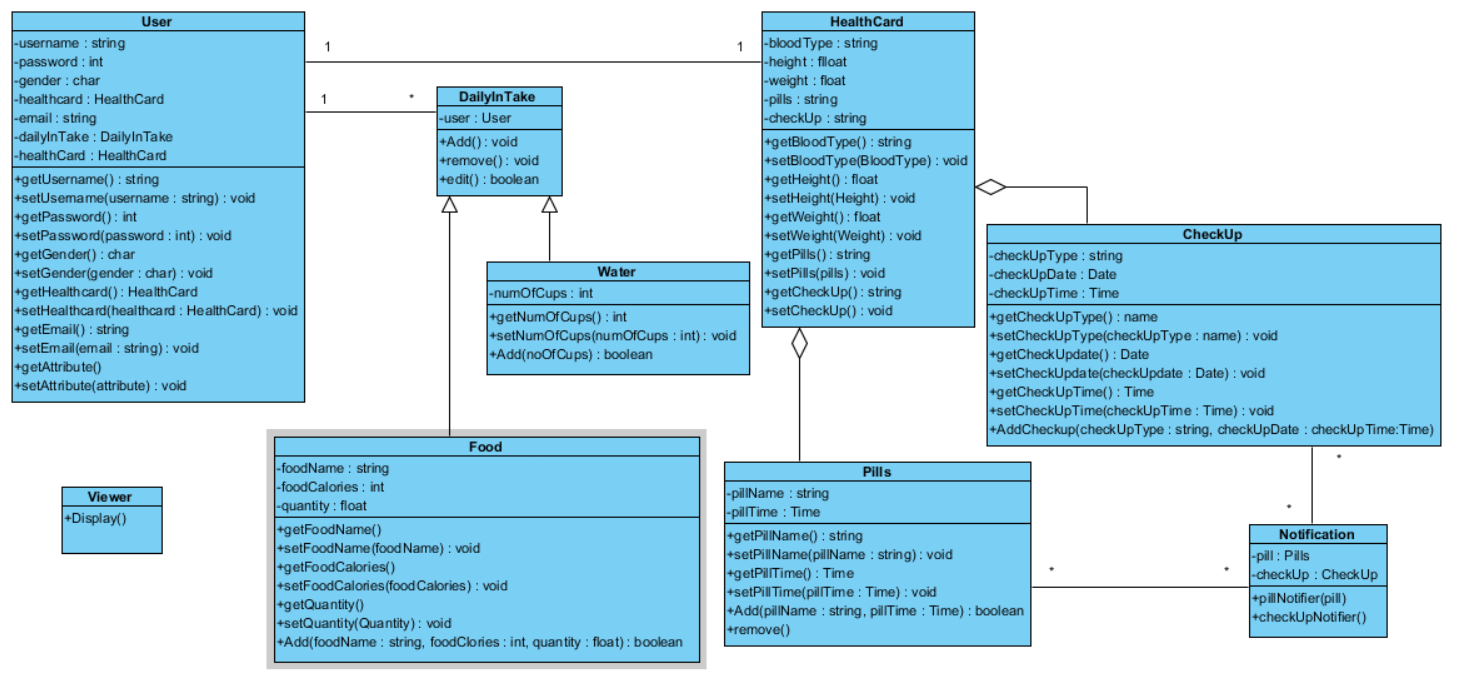
## I. System Decomposition

* **Provide and describe a figure that depicts the overall system decomposition.**
* **Read about “Software Coupling and Cohesion” Concepts. Reflect that in model**
* **The three following images from web are for purpose of explaining. First one is how model is designed, but there are little connections (so it is not so good).The other two ones have better connection. Consider the 3 to know how a good model should be.**
* **Remove these images from your report and draw your own system decomposition.**
* **System Decomposition**

****

## II. Class diagrams

* **You should provide your class diagram. In case on diagram is so complex, divide it to several ones of reasonable size or draw separate ones, each for one of the components on the system decomposition diagram.**
* **Class diagram is a static diagram and should not represent any dynamic flow of events.**
* **Put stereotypes of the classes to give more information. UML predefines some stereotypes like: <<interface>>, <<type>>, <<implementationClass>>, <<enumeration>>, etc. and you create your own also.**
* **Put Relationships between classes and the types of the relationships.**
* **Put multiplicity.**
* **Put relationship name (e.g. faculty "offer" course).**
* **Put attributes in the classes.**
* **Put functions & Put parameters.**
* **Put data types of each attributes and the parameters.**
* **Make sure to include all domain (entity), boundary and control classes needed to implement the system.**
* **Highly perfered: Each class has a corresponding interface**
  + **Let all objects parameters and returns be of interface type.**
  + **See Shopping Cart Case Study**
* **Following is Shopping Cart Component class diagram.**
* **Class Diagram**



**List down your classes and describe them**

| **Class ID** | **Class Name** | **Subsystem ID** | **Description & Responsibility** |
| --- | --- | --- | --- |
|  |  |  |  |

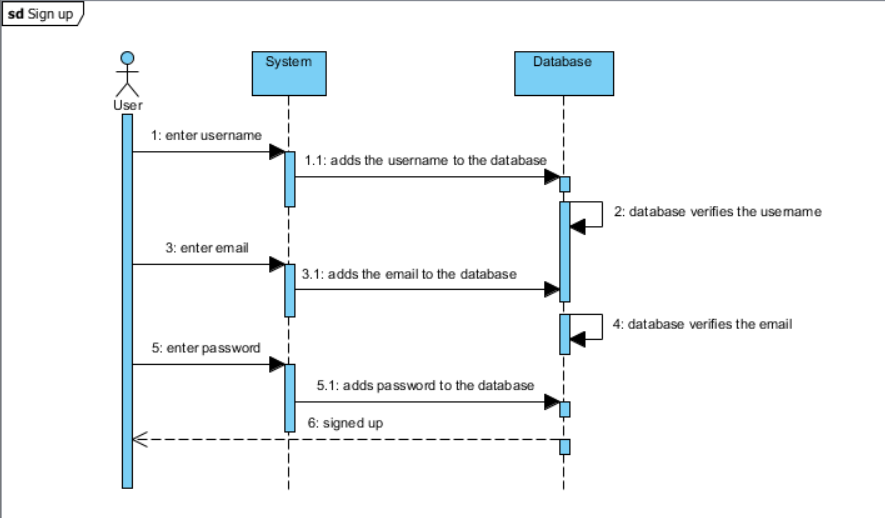
* **In the above table make sure that each class belongs to a subsystem.**
* **In the above table ALL classes should belong to subsystems. And each subsystem should at least contain one class.**

### Important Algorithm

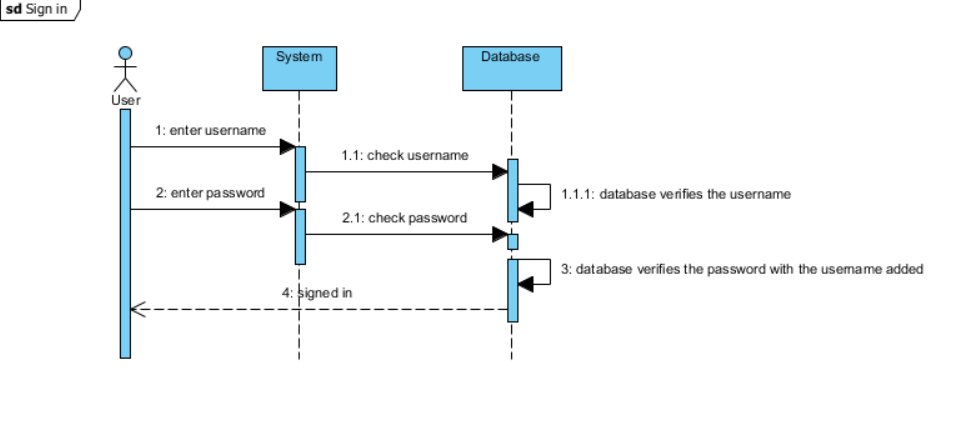
* **If any method in a class is implementing an algorithm (complex enough), then you should describe it here.**

## III. Sequence diagrams

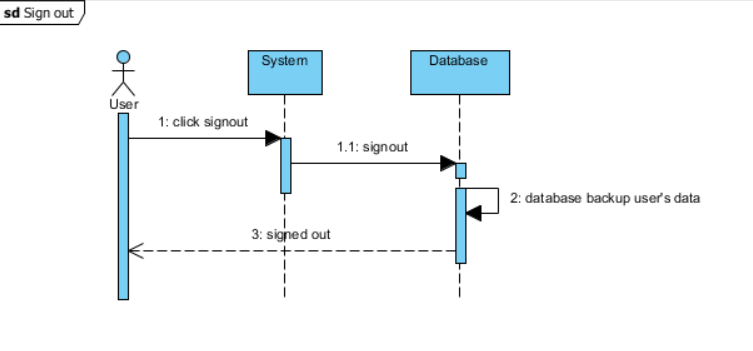
* **List Sequence diagrams for all requirements. Provide for each Sequence an ID.**
* **Usually each use case is represented by a sequence diagram or more.**
* **Overall, all the diagrams should represent all requirements and possible flows.**
* **Make sure that each object in the sequence diagram has a corresponding class in the class description table above. If not, it will be REJECTED.**
* **Put actual function calls with proper parameters and return types corresponding to class diagrams.**
* **Following are couple of examples for small / meduim examples. We expect such diagrams, however there is a missing thing in them. Most of calls don’t have parameters. Please always specify the parameters in the call, matching the class diagram.**
* **Sign up**

****

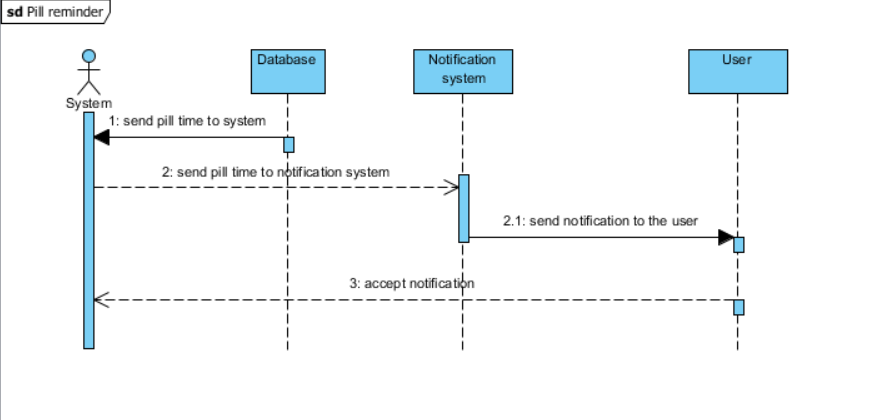
* **Sign in**



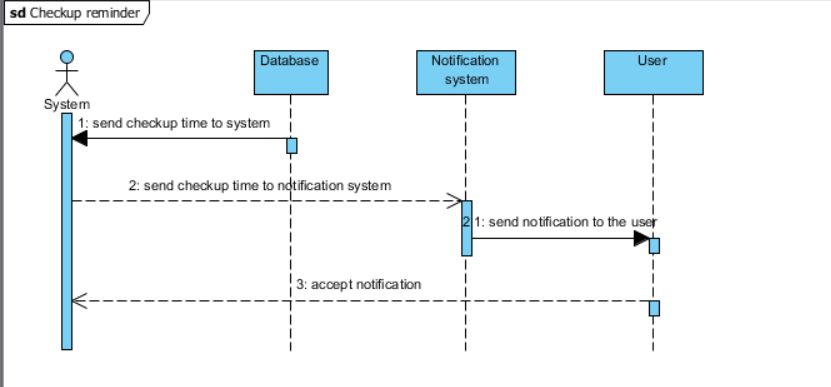
* **Sign out**



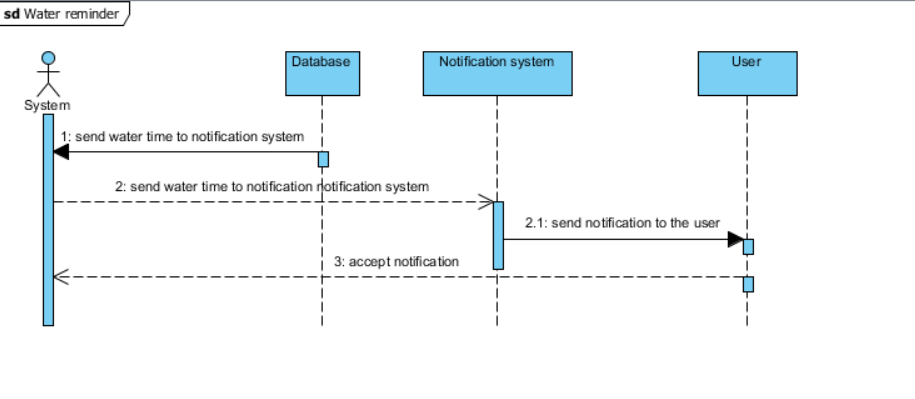
* **Pill reminder**



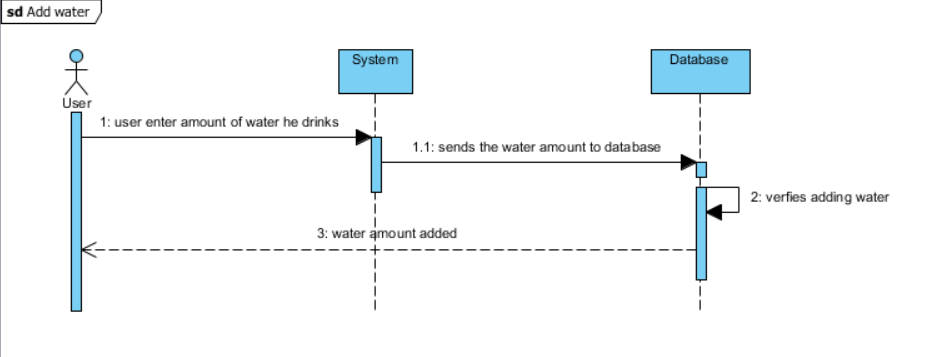
* **Checkup reminder**



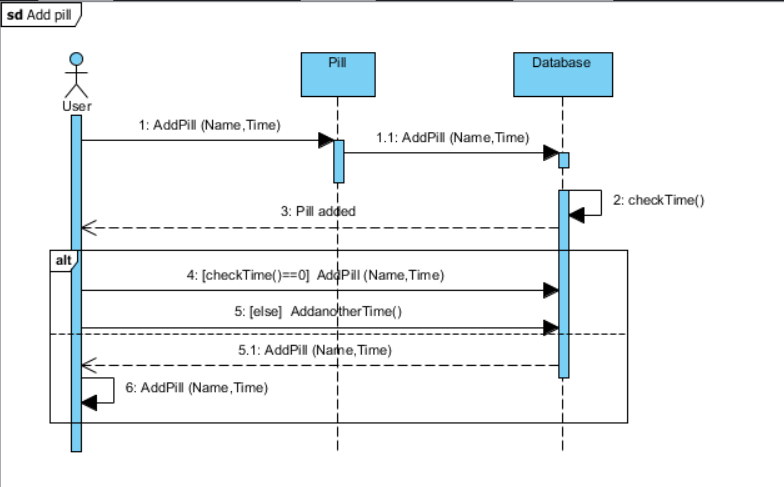
* **Water reminder**



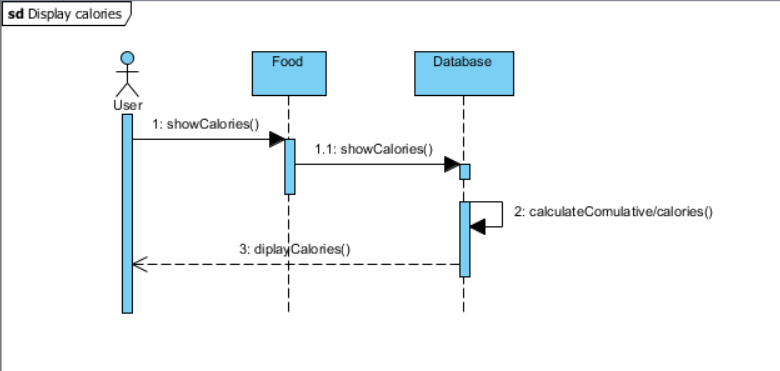
* **Add water**



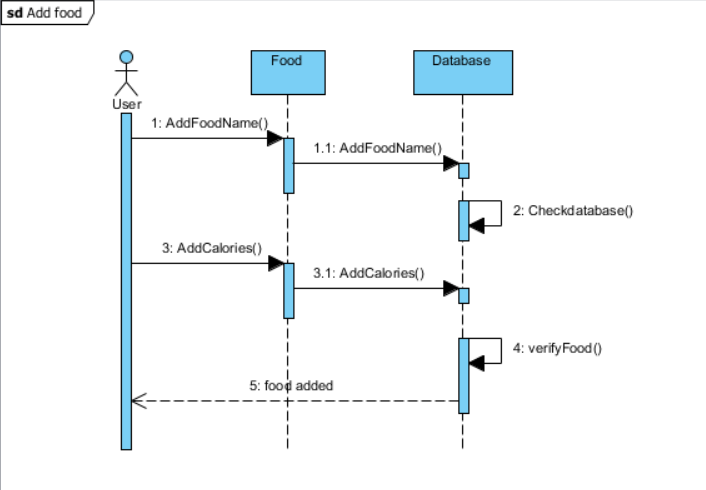
* **Add pill**



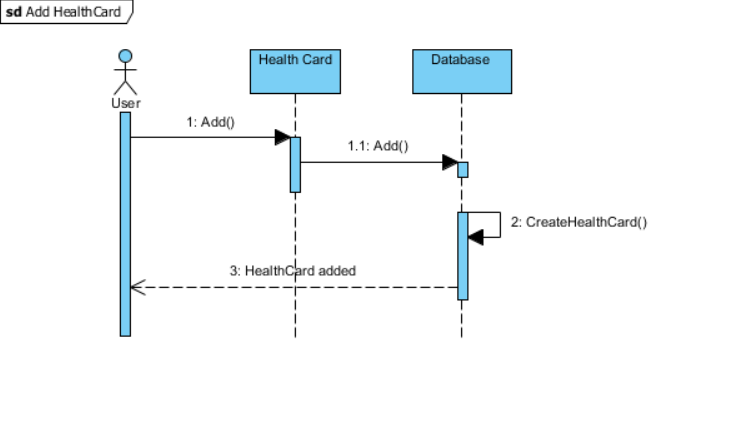
* **Calculate calories**



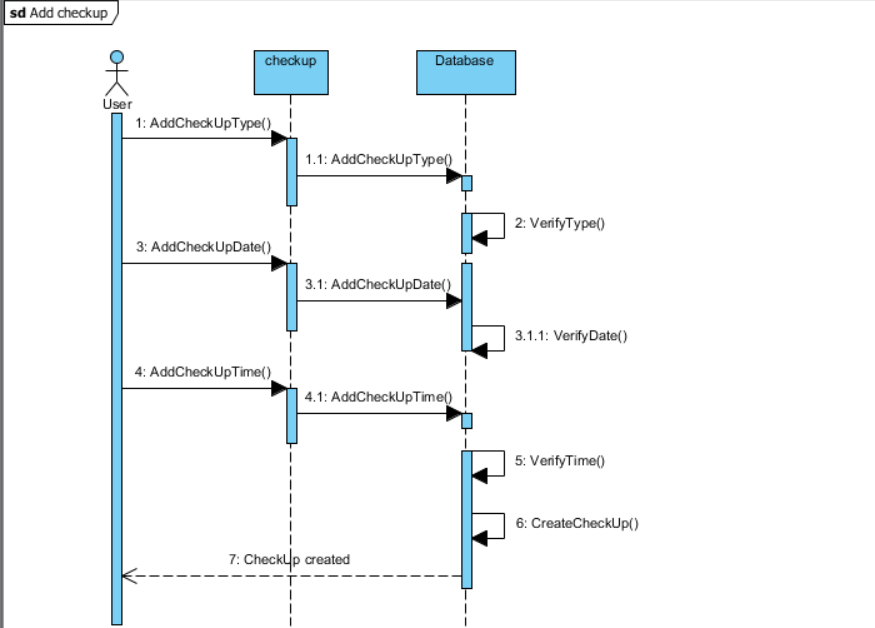
* **Add food**



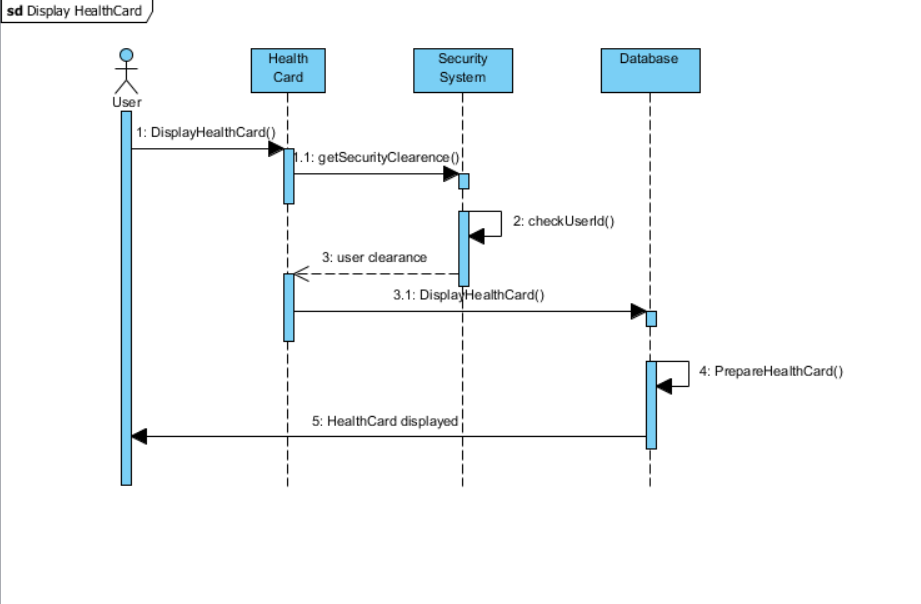
* **Add Health card**



* **Add checkup**



* **Health card**



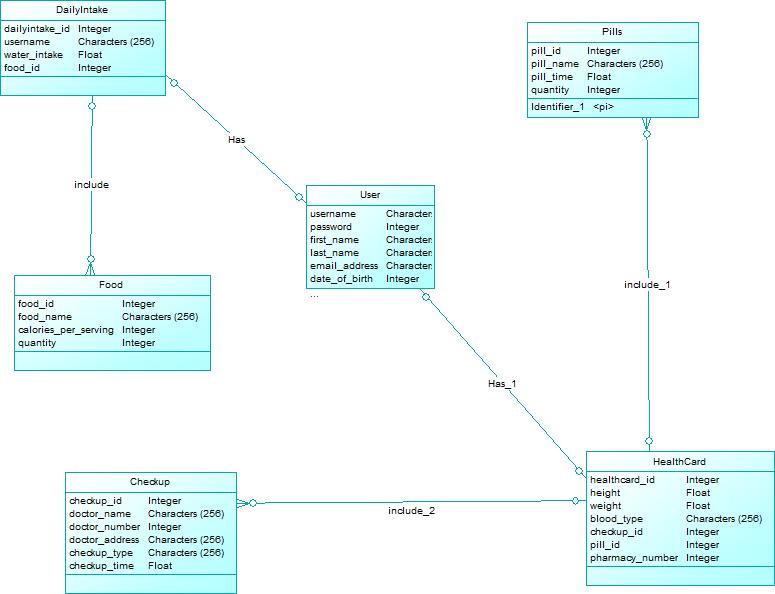
### Class - Sequence Usage Table

* **In this table, we will list EVERY class in class diagram and which sequences used this class diagram. This helps in avoiding either unused classes or extra classes appears in sequence diagrams. In "Overall used methods" section, put all functions appeared in all sequences. If this table was built in ignorance of actual class / sequence diagrams = REJECTED for whole document.**

| **Class Name** | **Sequence Diagrams** | **Overall used methods** |
| --- | --- | --- |
| E.g. Employee | 1, 3, 5 (means Seq Ids 1, 3, 5 used Employee class) | Save, GetData |
|  |  |  |

## IV. Physical Entity-Relationship Diagram

* **Provide the ERD Diagram**
* **All entity classes on the class diagram that need to be stored permanently, convert them and their relationships into ERD diagram.**
* **Don’t list any boundry or control entities!**
* **Following is an example of ERD**

****

## V. User Interface Design

* **Use a prototyping tool like** [**https://app.moqups.com**](https://app.moqups.com) **or a GUI builder to build your interface.**
* **Develop a prototype for each screen / page that your application will have and relate them to each other showing which one leads to which one.**
* **For each screen specify the buttons, menus, etc. that will be on it and their functions.**
* **An example is shown below.**
* **Screen 1 – Welcome screen**

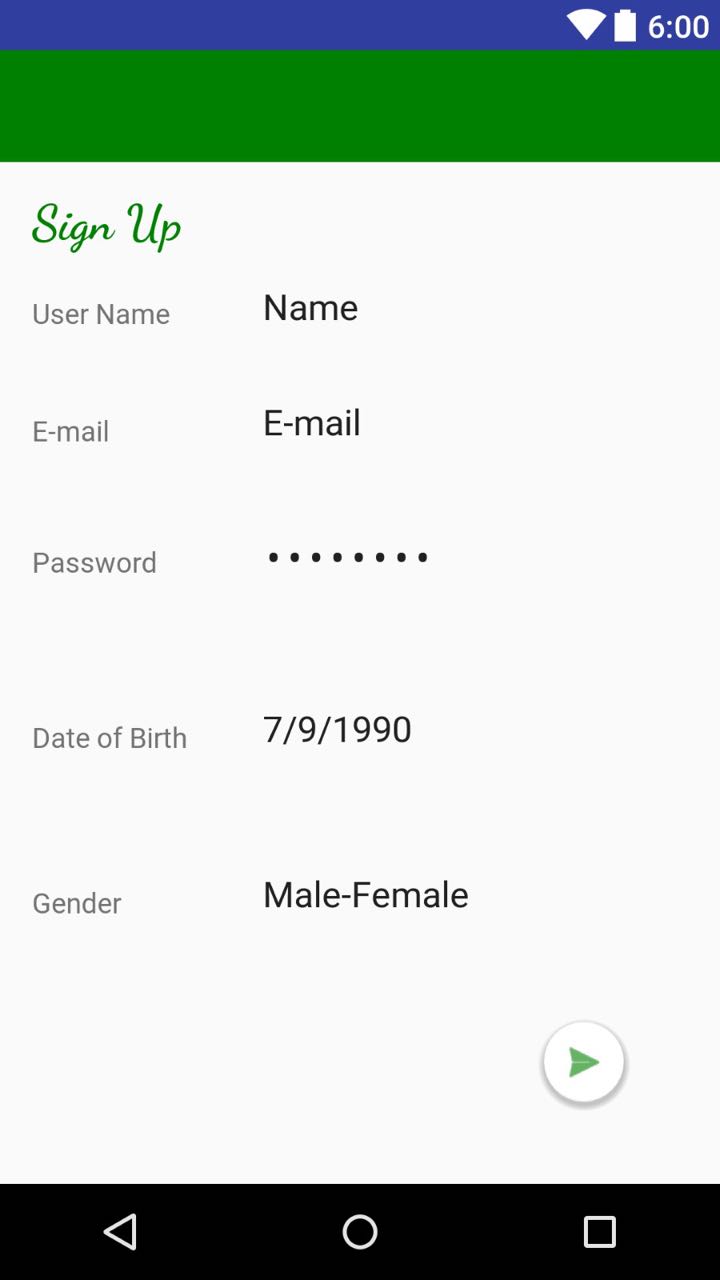
****

-

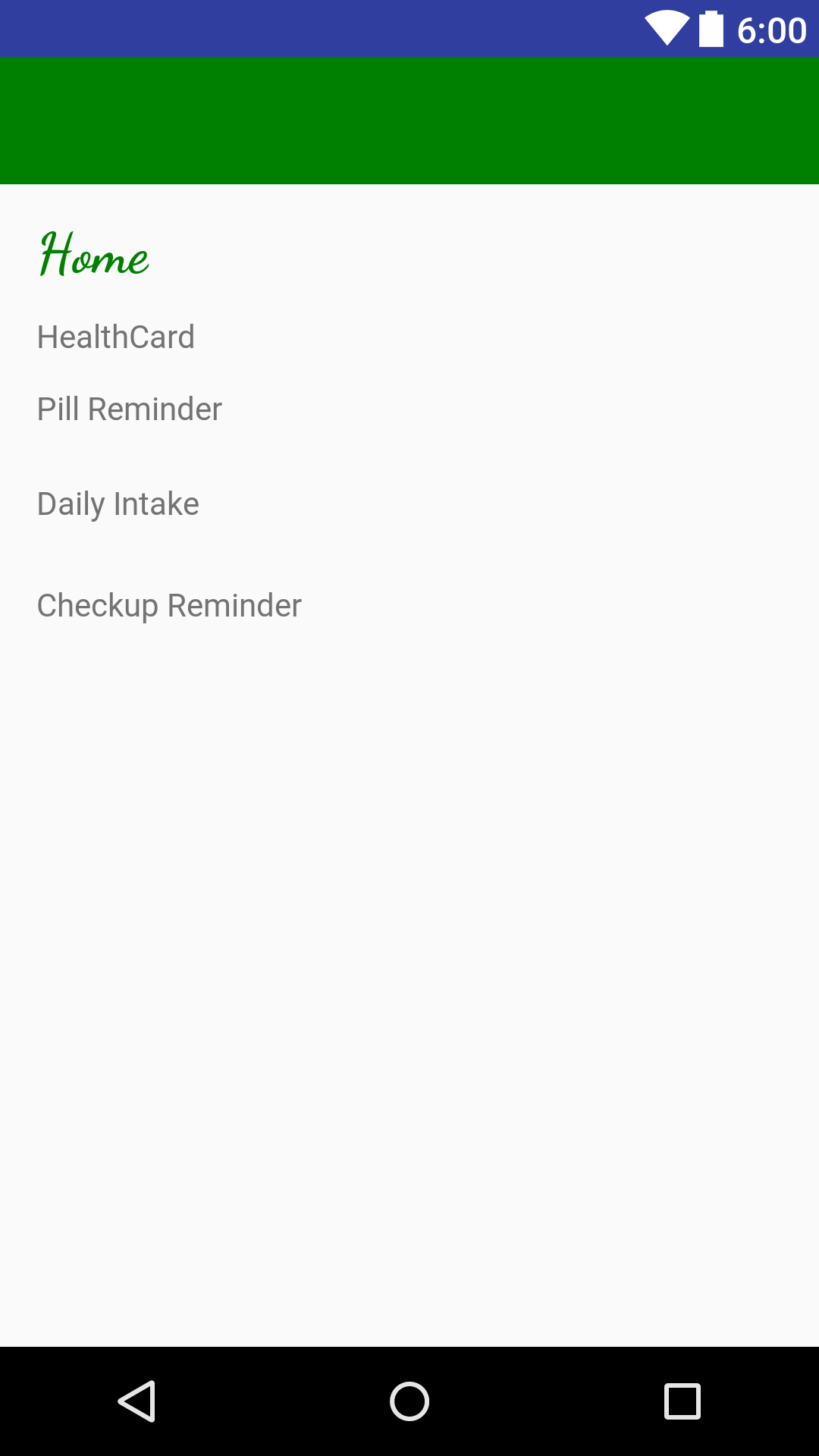
* **Screen 2 – Sign In**

****

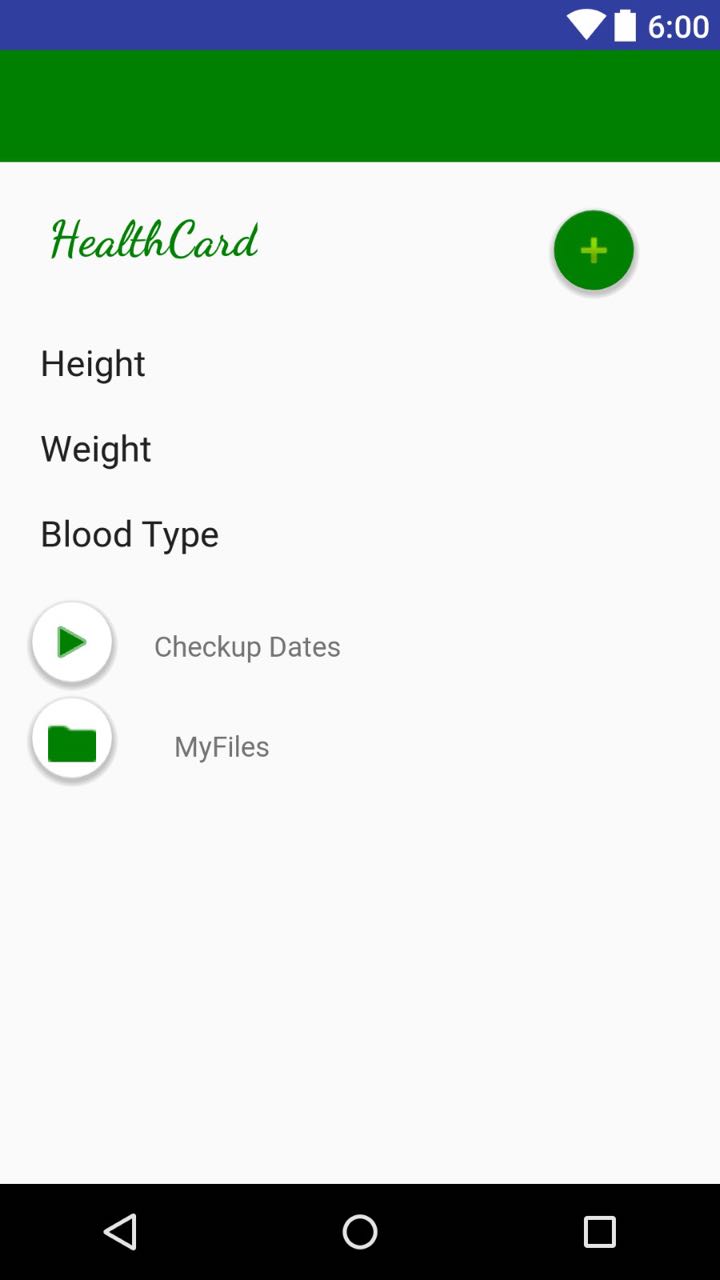
* **Screen3: Sign Up**



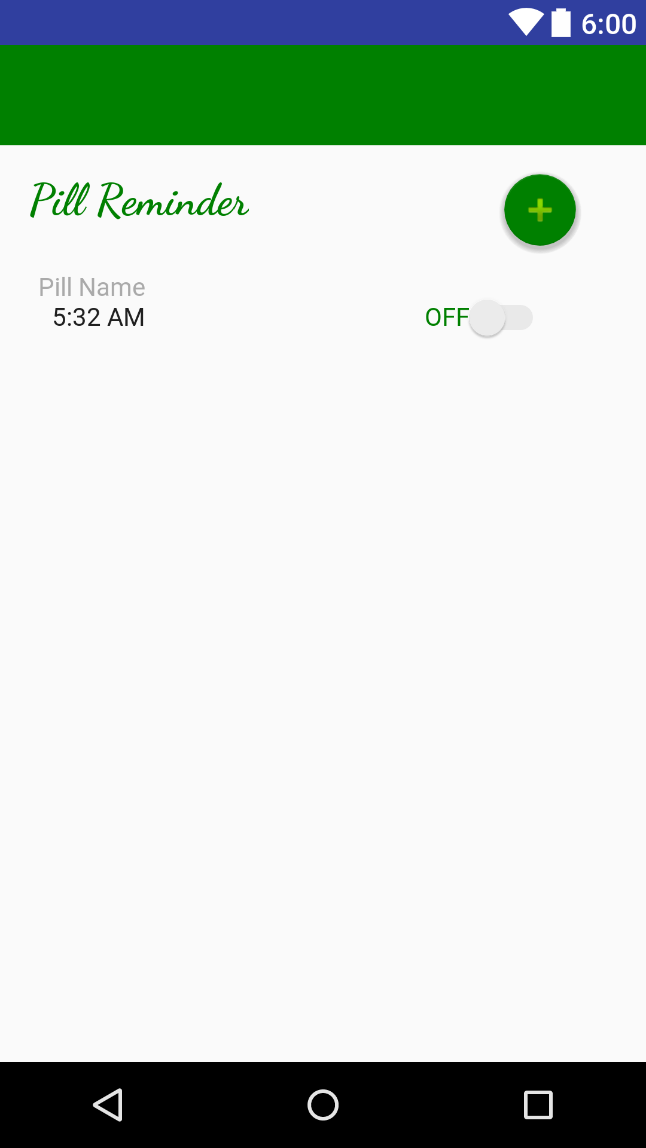
* **Screen 4: Home**



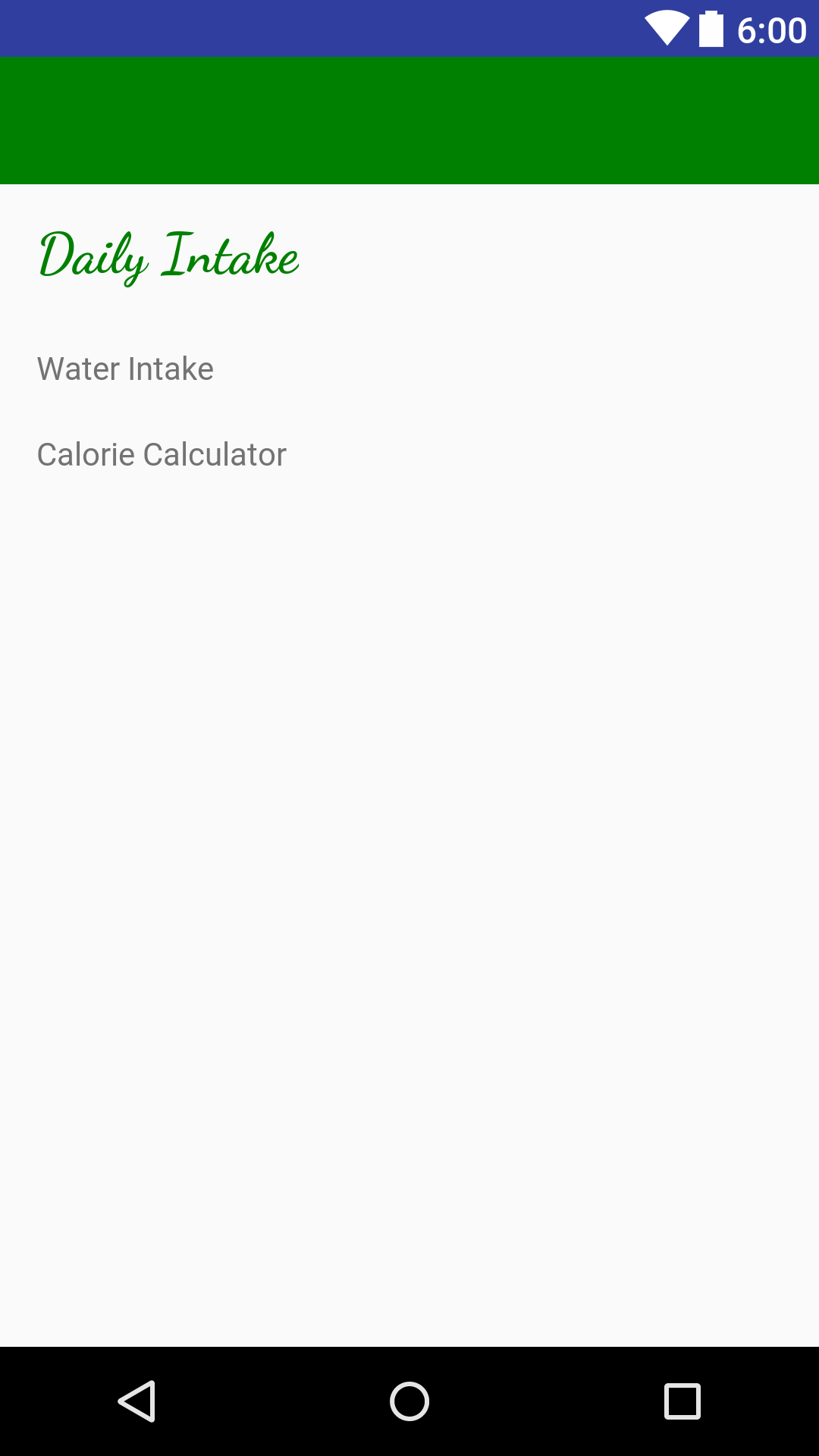
* **Screen5: Health card**



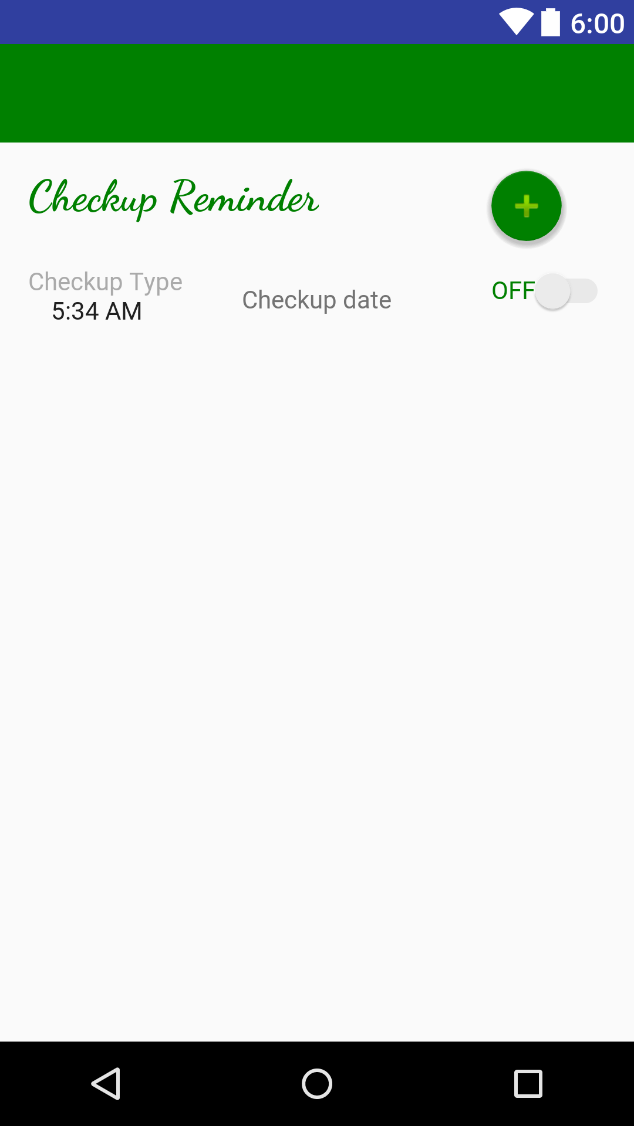
* **Screen6: Pill reminder**



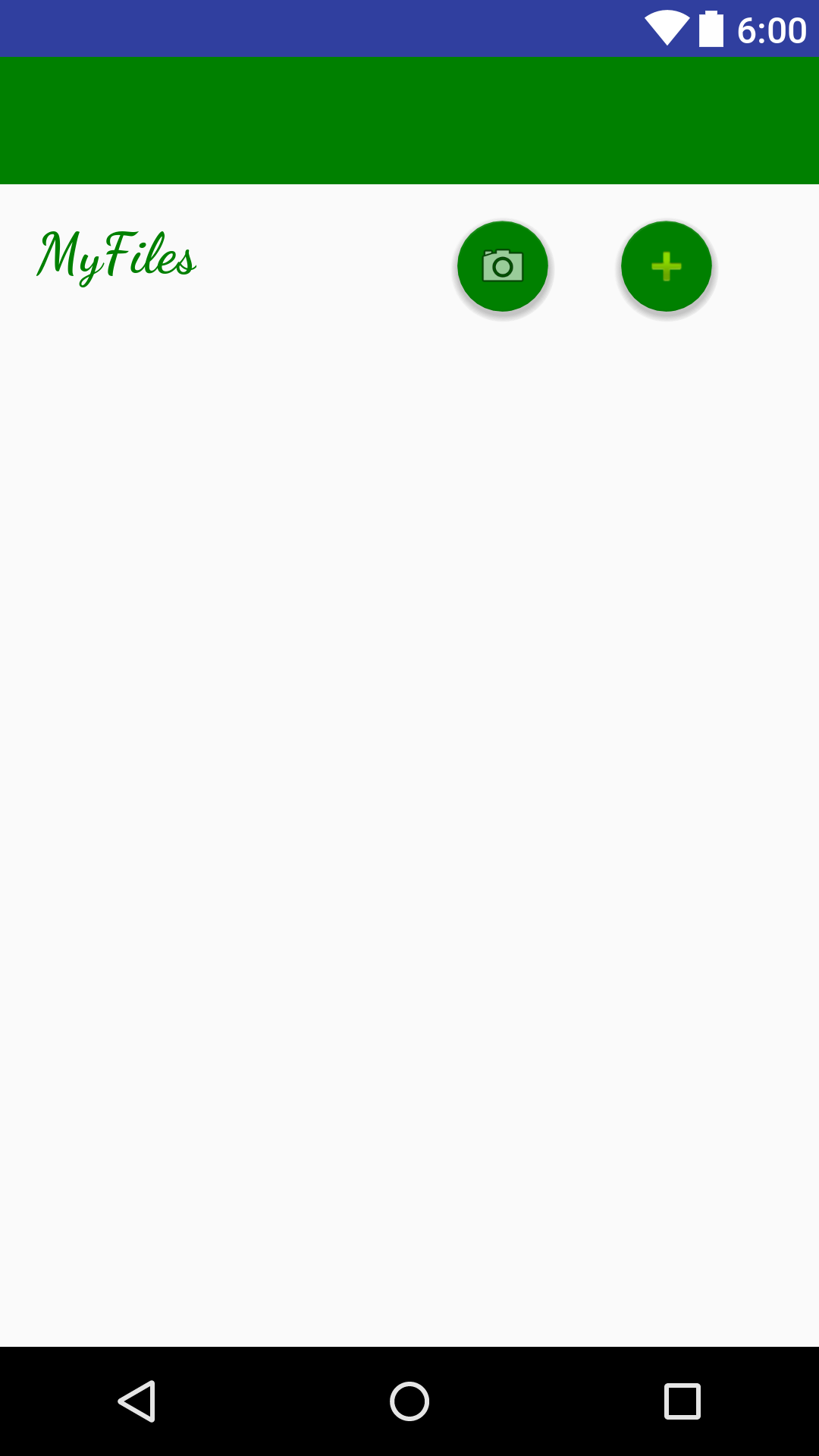
**Screen 7: Daily Intake**



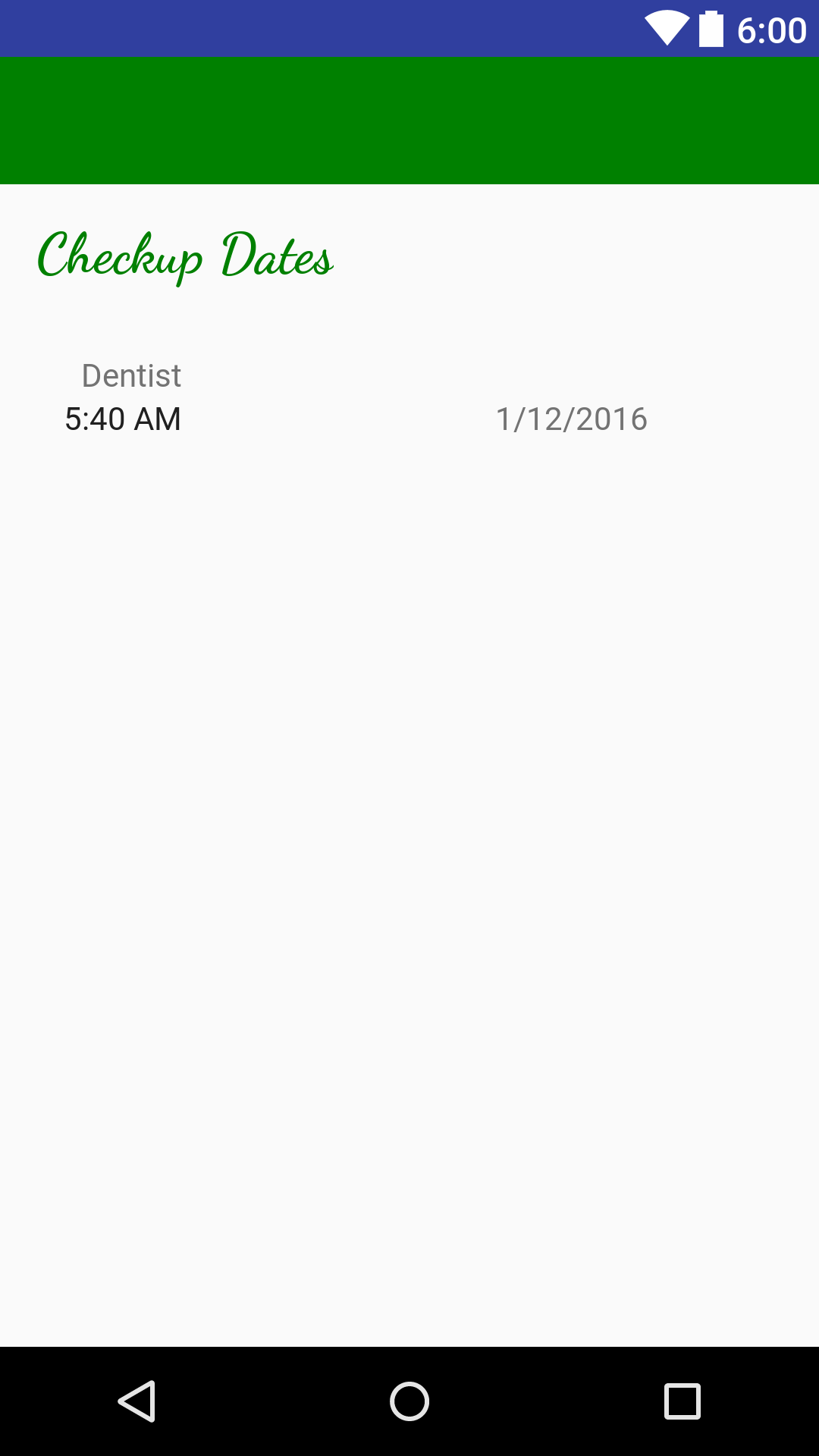
**Screen 8: Checkup Reminder**



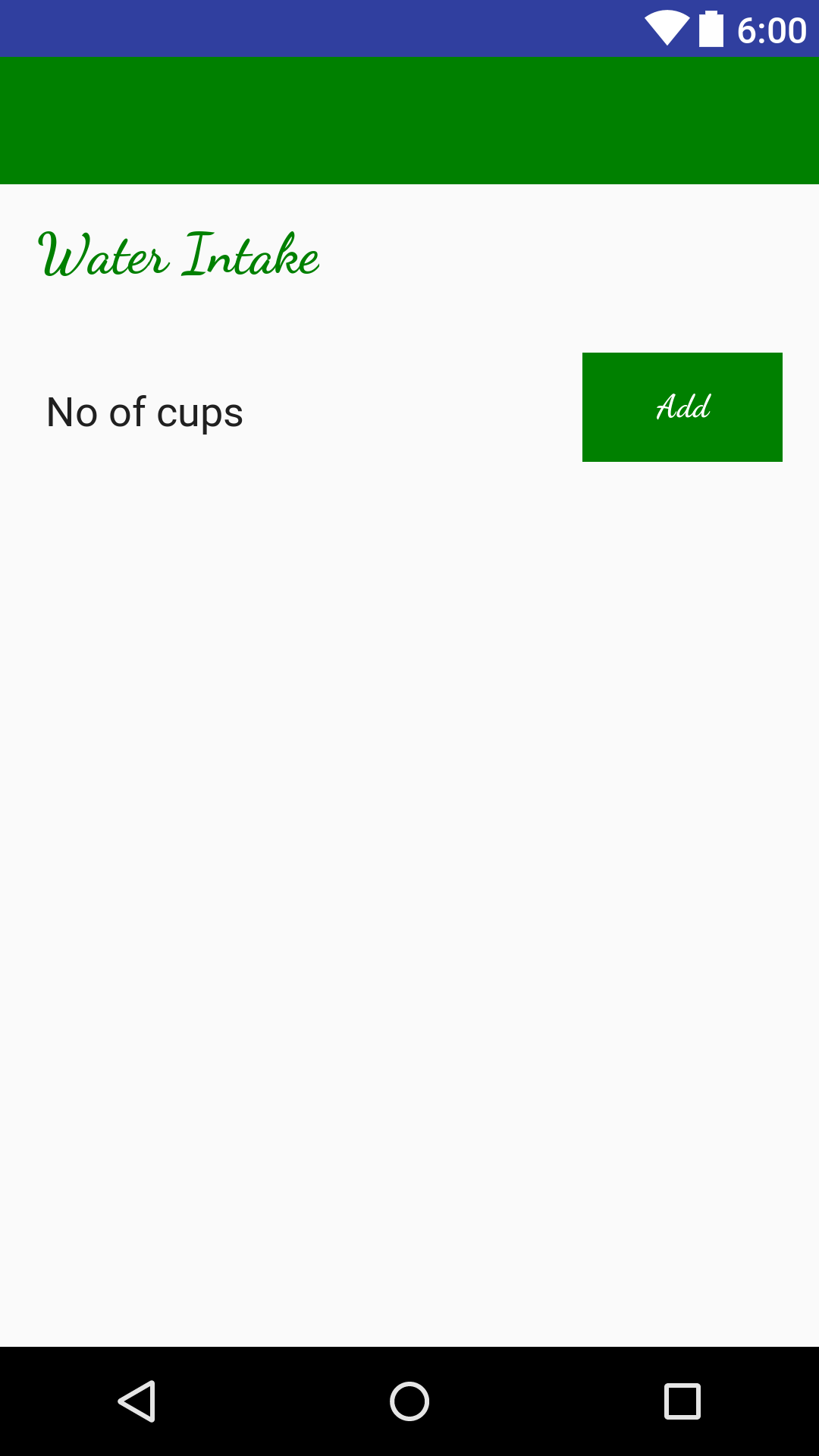
**Screen 9: My Files**



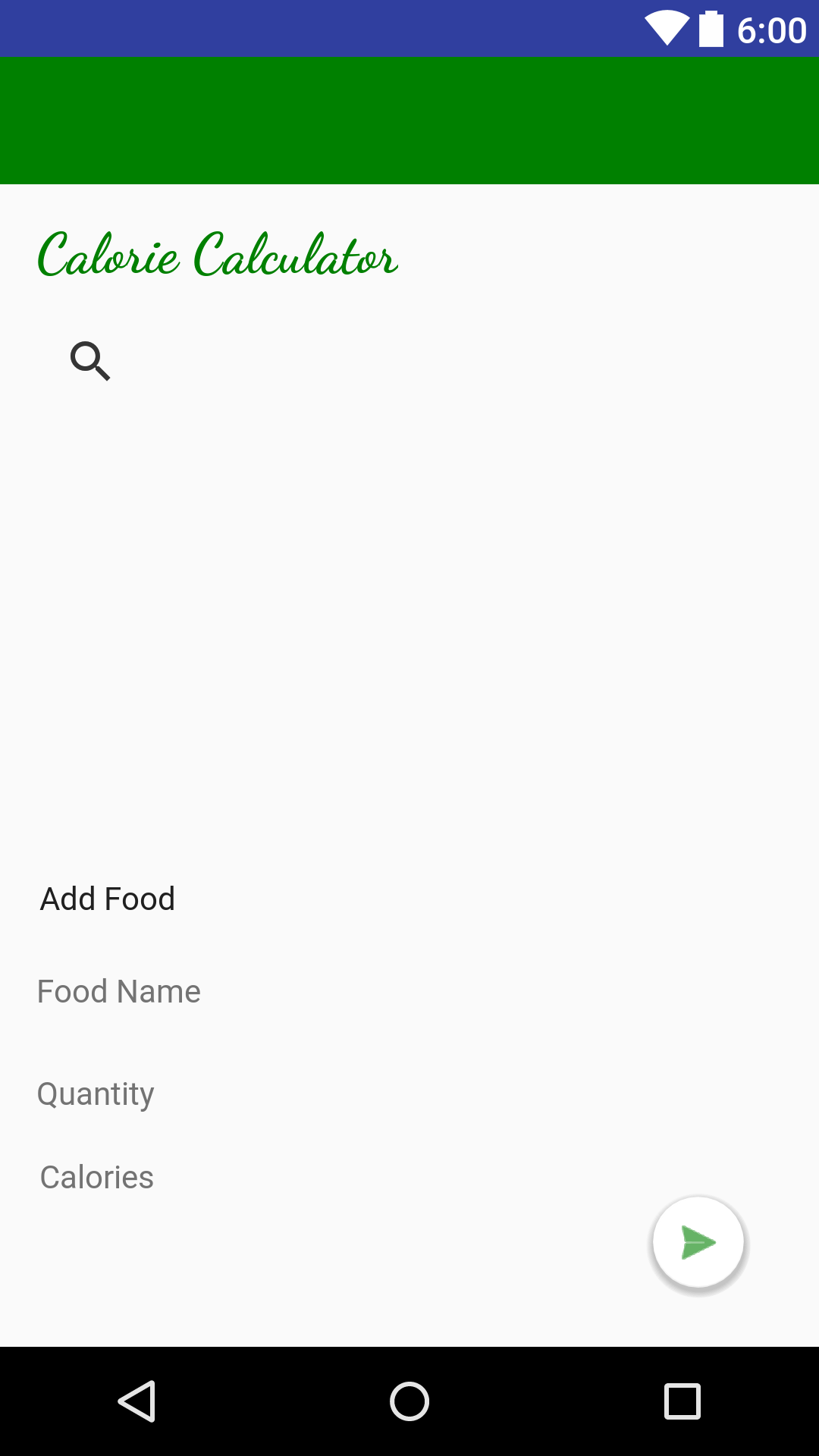
**Screen 10: Checkup Dates**



**Screen 11: Water Intake**



**Screen 12: Calory Calculator**



* **Navigation tree: Login Screen**

**|**

* **Navigation tree: Main Screen**

## VI. Bonus: State Diagram and OCL

* **Choose a non-trivial entity that has different states and develop a state diagram for it.**
* **Choose 5 functional requirements that cannot be represented by class and sequence diagrams and develop OCL constraints for them.**

# Ownership Report

* **Remove the following notes and any red notes**
* **For every item in this document, write the owners. If someone is owner of something, s/he understands it 100.%**
* **Team leader must verify the table with the team members.**

|  |  |
| --- | --- |
| **Item** | **Owners** |
|  |  |
|  |  |

# References

* http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs1/design.pdf

# Authors

* Mostafa Saad and Mohammad El-Ramly