

**Mobile health screening platform for degenerative
neurological disorders**



Politi Vasiliki

Thesis Committee:

Balas Constantinos, Associate Professor (Supervisor)

Mania Aikaterini, Associate Professor

Chalkiadakis Georgios, Assistant Professor

Chania, July 2014

Acknowledgements

First of all, I would like to thank my advisor Balas Constantinos for his help and trust. Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Aikaterini Mania and Prof. Chalkiadakis Georgios for their participation in the presentation and the evaluation of this diploma thesis.

Finally I would like to thank all those who provided me their support and encouragement, in order to make this thesis possible, especially my friends and collaborators.

Abstract

The new and dynamic field of mHealth - the use of mobile technology applications for healthcare - could improve the well-being of people around the world. Mobile applications can lower costs and improve the quality of healthcare as well as shift behavior to strengthen prevention, all of which can improve health outcomes over the long term.

Technologies of mHealth field are a valuable partner in health care's shift towards a delivery model that is patient-centered and value-based.

A broad category of brain diseases is Dementia, which causes long term loss of the ability to think and reason clearly that is severe enough to affect a person's daily functioning. The most common form of Dementia is Alzheimer's disease (75%). Although Alzheimer's disease develops differently for every individual, there are many common symptoms. Early symptoms are often mistakenly thought to be 'age-related' concerns, or manifestations of stress.

Neurology cognitive tests used for symptoms recognition to such degenerative disorders are generally applied by paper-and-pencil until now. These kinds of tests implemented until now on PCs have typically unfriendly interface and do not take people's capabilities into consideration.

Our aim was to develop a mHealth screening platform that combines various neurological tests that will help the diagnosis of early dementia and Alzheimer disease by easy performed self-testing applications, provided by a friendly GUI. The application is composed of the best 3 tests chosen from the paper-and-pencil standard tests, Clock drawing test (CDT), Mini Cog test (MCT) and Montreal Cognitive Assessment test (MoCA).

This thesis is important for being one of the first mobile applications for degenerative neurological disorders self-testing that is compatible for tablets and smart-phones regardless witch operating system is running.

Περίληψη

Ο νέος και δυναμικός τομέας της «κινητής υγείας» (mHealth) - χρήση των εφαρμογών κινητής τεχνολογίας στον τομέα της υγείας - θα μπορούσε να βελτιώσει την ποιότητα ζωής των ανθρώπων σε όλο τον κόσμο. Οι εφαρμογές αυτού του τύπου μπορούν να μειώσουν τα κόστη περίθαλψης, να βελτιώσουν την ποιότητα παροχών υγείας, καθώς και να ενισχύσουν την τάση για την πρόληψη ασθενειών, οπότε μακροπρόθεσμα να βελτιωθούν τα αποτελέσματα του τομέα.

Οι τεχνολογίες του τομέα «κινητής υγείας» αποτελούν πολύτιμο συνεργάτη στην τάση προς ένα μοντέλο υγείας που θα έχει επίκεντρο τον ασθενή βασισμένο σε ποιοτικές αξίες.

Μια ευρεία κατηγορία των νόσων του εγκεφάλου είναι η άνοια, η οποία προκαλεί μακροπρόθεσμα την απώλεια της ικανότητας σκέψης και λόγου με σαφήνεια και είναι αρκετή έτσι ώστε να επηρεάσει σοβαρά την λειτουργικότητα του ατόμου στην καθημερινότητά του. Η πιο κοινή μορφή άνοιας είναι η νόσος του Αλτσχάιμερ (75%). Αν και η νόσος του Αλτσχάιμερ εξελίσσεται διαφορετικά για κάθε άτομο, υπάρχουν πολλά κοινά συμπτώματα. Τα πρώιμα συμπτώματα είναι συχνά λανθασμένα καθώς συνηθίζεται να σχετίζονται λανθασμένα με προβλήματα λόγω ηλικίας ή εκδηλώσεις στρες.

Τα Νευρολογικά γνωστικά τεστ για αναγνώριση συμπτωμάτων τέτοιου τύπου εκφυλιστικών ασθενειών εφαρμόζονται γενικά μέχρι τώρα στην έντυπη μορφή τους. Παρόμοια τεστ που έχουν υλοποιηθεί στους υπολογιστές έχουν συνήθως “μη φιλική” διεπαφή με τον χρήστη και δεν λαμβάνουν υπόψη όλες τις δυνατότητες των ατόμων.

Στόχος μας ήταν να δημιουργήσουμε μια πλατφόρμα που συνδυάζει διάφορα τεστ που θα βοηθήσει στη διάγνωση της πρώιμης άνοιας και της νόσου του Αλτσχάιμερ μέσα από μια εύκολα χρησιμοποιούμενη εφαρμογή αυτοελέγχου που παρέχεται σε «φιλικό» περιβάλλον . Η εφαρμογή αποτελείται από τα 3 αποτελεσματικότερα τεστ των τυποποιημένων έντυπων, το Ρολόι, (CDT), το Mini Cog (MCT) και το MoCa.

Η διπλωματική αυτή είναι σημαντική γιατί αποτελεί μια από τις πρώτες υλοποιήσεις εφαρμογής για αυτοέλεγχο σε νευρολογικές εκφυλιστικές ασθένειες και είναι συμβατή για ταμπλέτες και έξυπνα τηλέφωνα υποστηρίζοντας όλα τα είδη των λειτουργικών συστημάτων.

Table of contents

Chapter 1 – Preface

1.1 - Introduction	10
1.2 - Background	11
1.3 - Thesis contribution	11
1.4 - Thesis Outline	11

Chapter 2 – Medical part

2.1 - Degenerative nerve diseases	12
2.2 - Neurological tests	15
2.2.1 - Clock drawing test	15
2.2.2 - Mini Cog Test	17
2.2.3 - MoCa Test	18

Chapter 3 – Technology part

3.1 Mobile applications approach	20
3.1.1 Native applications	20
3.1.2 Web applications	23
3.1.3 Hybrid applications	25
3.1.4 Comparing different approaches	28
3.2 Development App tools	29
3.2.1 PhoneGap	29
3.2.2 Cordova	31
3.2.3 Eclipse	34
3.3 Programming Languages	35
3.3.1 HTML5	35
3.3.2 CSS3	37
3.3.3 AngularJS	38
3.4 Application Technical Analysis	45

Chapter 4 – Our Approach

4.1 Methodology	52
4.2 Test and Interfaces	53

4.2.1 Login screen	53
4.2.1 Dashboard	54
4.2.1 Previous Tests	55
4.2.4 Clock drawing test	56
4.2.5 Mini Cog test	59
4.2.6 MoCa test	61
 Chapter 5 – Conclusions	
5.1 Usability	67
5.2 Future Work	68
 Appendices	
Appendix A - Abbreviations	70
Appendix B –Models Sample Source code	71
Appendix C - Bibliography	73

List of figures

2.1 - Computer based tests	13
2.2 - Cognitive Impairment from CDT	15
2.3 - No Cognitive Impairment from CDT	16
2.4 - Mini-Cog Scoring Algorithm	17
2.5 - MoCa Scores	18
2.6 - MoCa Test Sample	19
3.1 - Features overview of Native Apps Programming	21
3.2 - Features overview of Web Apps Programming.	24
3.3 - Examples of Hybrid Apps	27
3.4 - Comparison between mobile application approaches	28
3.5 - Schematic presentation of mobile apps	28
3.6 - PhoneGap Schematic Overview	29
3.7 - Cordova Schematic overview	31
3.8 - Cordova Architecture.	32
3.9 - Cordova UI.	32
3.10 - Supported Features by Cordova	33
3.11 - Initialization of AngularJs	39
3.12 - AnjularJs Update Schema	40
3.13 - AnjularJs Data Binding.	44
3.14 - Cordova/ PhoneGap Architecture Diagram	46
3.15 - Cordova output folder structure	47
3.16 - Project flow diagram	47
3.17 - Folder structure of code	48
3.18 - Android emulation	51
4.1 - Application's Login screen	53
4.2 - Prompt screen for new user	53
4.3 - Dashboard	54
4.4 - Tests history	55
4.5 - Prompt to fill numbers in CDT	57
4.6 - Correct filled numbers in CDT	57
4.7 - Prompt to wriggle clockwise in CDT	58
4.8 - Correct time in CDT	58
4.9 - Words to remember in Mini Cog	59
4.10 - Showing words previously given in Mini Cog	60
4.11 - Selecting words previously given in Mini Cog	60
4.12 - Cube prompt to copy in Moca	61

4.13 - Preceded drawing of cube in MoCa	62
4.14 - Fully copied cube in MoCa	62
4.15 - Picture to remember in MoCa	63
4.16 - Past picture to choose in MoCa	63
4.17 - Prompting to name animals in MoCa	63
4.18 - Correct naming in MoCa	64
4.19 - Long memory part 1 in MoCa	64
4.20 - Attention subtest in MoCa	65
4.21 - Category choice in MoCa	65
4.22 - Long memory part 2 in MoCa	66
4.23 - Subtraction part in MoCa	66

Chapter 1 - Preface

1.1 Introduction

mHealth has been defined as "medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices." The power and reach of mobile communications offers great versatility and utility to enable provision of high-quality, low-cost health services.

2013 was a big year for the adoption of mobile medical applications by clinicians and consumers. It also saw the beginning of convergence between devices and apps used by clinicians and those used by consumers.

Downloadable health care-related apps for smart devices are a rapidly growing component of mHealth. Substantial growth in this market is projected. Estimates expect the medical apps industry to grow by 23 percent annually over the next five years. Mobile application users who downloaded at least one mHealth app onto their smartphone were estimated at 127 million in 2011 and are projected to double to 247 million in 2015. 30% of mobile apps target health professionals, facilitating such things as access to patient data, patient consultation and monitoring, diagnostic imaging, and pharmaceuticals information.

The increasing adoption and use of mobile technologies is disrupting the healthcare industry. This phenomenon has created innovative ways, channels and tools to deliver healthcare cost-effectively even in the remotest of places. A quick search for "health" will result in over 43,000 apps. The number of app downloads demonstrates that there is a high demand for such applications. Mobile apps are radically changing the way doctors and patients interact and approach health care.

1.2 Background

Technologies are used in research and education, knowledge transfer, social support, service delivery and health promotion. Mobile technology is increasingly used in telemedicine, wireless monitoring of health outcomes in disease management and delivery of health interventions. Mobile phones are emerging as an important method of encouraging better nurse-patient communication and are estimated to increase in use and application over coming years.

There is currently little published research in neurological self-tests and more evidence is needed for this kind of applications. Although this factor evidence for the application of mobile technology in the care of older people is just emerging and will almost certainly develop in coming years.

1.3 Thesis contribution

Our aim is to develop a mHealth screening platform (compatible with all kind of OS) that combines self-tests of cognitive screening (doctor's absence) and friendly user interface.

1.4 Thesis Outline

In **Chapter 2** we analyze the medical fields covered by this thesis. In **Chapter 3** we present the technology needed to implement the mobile application. **Chapter 4** shows screenshots of the application covering all the neurological tests. Finally conclusions are analyzed in **Chapter 5**.

Chapter 2 – Medical part

2.1 Degenerative nerve diseases

Degenerative nerve diseases affect many of body's activities, such as balance, movement, talking, breathing, and heart function. Many of these diseases are genetic. Sometimes the cause is a medical condition such as alcoholism, a tumor, or a stroke. Other causes may include toxins, chemicals, and viruses. Sometimes the cause is not known.

Degenerative nerve diseases include:

- ✓ Alzheimer's disease
- ✓ Amyotrophic lateral sclerosis
- ✓ Friedreich's ataxia
- ✓ Huntington's disease
- ✓ Lewy body disease
- ✓ Parkinson's disease
- ✓ Spinal muscular atrophy

Degenerative nerve diseases can be serious or life-threatening. It depends on the type. Treatments may help improve symptoms, relieve pain, and increase mobility but most of them have no cure.

Dementia isn't a specific disease. Instead, dementia describes a group of symptoms affecting thinking and social abilities severely enough to interfere with daily functioning.

Many causes of dementia symptoms exist. Alzheimer's disease is the most common cause of a progressive dementia.

Dementia indicates problems with at least two brain functions, such as memory loss and impaired judgment or language, and the inability to perform some daily activities such as paying bills or becoming lost driving.

Dementia can make you confused and unable to remember people and names. You also may experience changes in personality and social behavior.

In the past couple of years, highly precise and reliable instruments have become available to screen for mild cognitive dysfunction in general and early presymptomatic dementia in particular. They are computerized neurocognitive test batteries. They are derived from the PC-based neuropsychological batteries that have been used for 25 years in military and aerospace medicine, the pharmaceutical industry, and industrial and sports medicine.

These tests include:

Computerized Batteries for Dementia Screening	
Battery	Source
Cantab-PAL	www.bioportfolio.com/cantab.html
CNS Vital Signs	www.cnsvs.com
CogScreen	www.cogscreen.com
CogState	www.cogstate.com
MicroCog	Psychological Corporation

Figure 2.1 – Computer based tests

All test batteries exhibit some of the strengths of computerized cognitive testing: standardization of administration and stimulus presentation, accurate measures of response latencies, automated comparison in real-time with an individual's prior performance as well as with age-related norms, and efficiencies of staffing and cost.

When computerized neurocognitive testing is compared head-to-head to conventional neuropsychological batteries, the results are comparable. Similar tests yield similar results. Each method has advantages and

disadvantages. Examination by a neuropsychologist is more flexible, more comprehensive, and more sensitive to subjective issues like the patient's mood, his level of motivation, and the possibility of exaggerated or invalid responding.

Computerized testing is quicker and less expensive. It is also more sensitive to MCI because most computerized tests are timed, and timing is usually done in milliseconds.

2.2 Neurological tests

2.2.1 Clock drawing test (CDT)

The clock-drawing test is a simple tool that is used to screen people for signs of neurological problems, such as Alzheimer's and other dementias. It is often used in combination with other, more thorough screening tests (like MoCa and Mini-Cog) but even when used independent it can provide helpful insight into a person's cognitive ability.

The clinician gives the person a blank piece of paper it and asks to draw a clock that shows a time (most of the times 10 minutes after 11).

There are as many as 15 different ways to score this test. Some are quite elaborate and involve awarding points for inclusion of every number, correctly ordered numbers, two clock hands, drawing the correct time, and for each of the correct numbers placed in the four quadrants. As many as five, 10, or 20 points can be involved in some of the different scoring methods. However, a study published in the January 2012 Danish Medical Journal outlines research that compared five of the most common ways to score the test and their conclusion was that the easiest scoring method provided results that were just as accurate as the more complicated scoring methods.

The easiest method identifies 4 features in this test that are important in successfully identifying problems in cognition. These are described as critical clock-drawing errors and include present or absent of the circle, missing numbers or number substitutions and wrong time. Score 0 indicates dementia and might be caused of the refusal or the inability to draw even the circle of the clock.

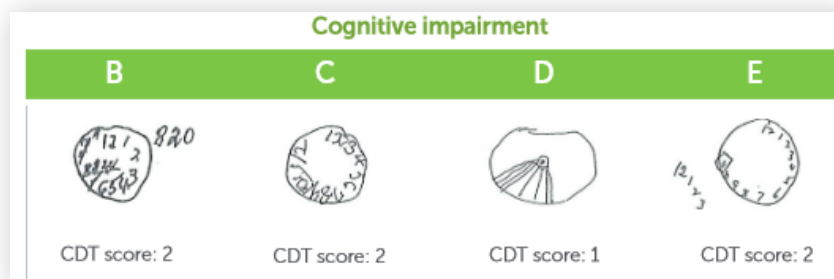


Figure 2.2. – Cognitive Impairment from CDT

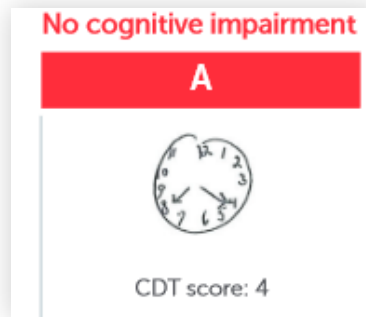


Figure 2.3 – No Cognitive Impairment from CDT

2.2.2 Mini-Cog Test

The Mini-Cog is a rapid screening test for Alzheimer's disease. Unlike other popular Alzheimer's tests that measure several aspects of cognition, such as the MMSE and the MoCA, the Mini-Cog measures only two: short-term recall and clock drawing test. Despite that, the Mini-Cog is extremely accurate at predicting whether someone has dementia.

First, the person is asked to repeat three unrelated words (chosen by clinical researchers). Then the person is asked to do the Clock drawing test analyzed previously. Finally, the person is asked to recall the three words.

Scoring of the Mini-Cog is simple as well. A person is scored as demented if they recall none of the three words, or if they recall one or two of the three words and draw an abnormal clock. Similarly, a person is scored as non-demented if they recall all three words or if they recall one or two of the three words but draw a normal clock.

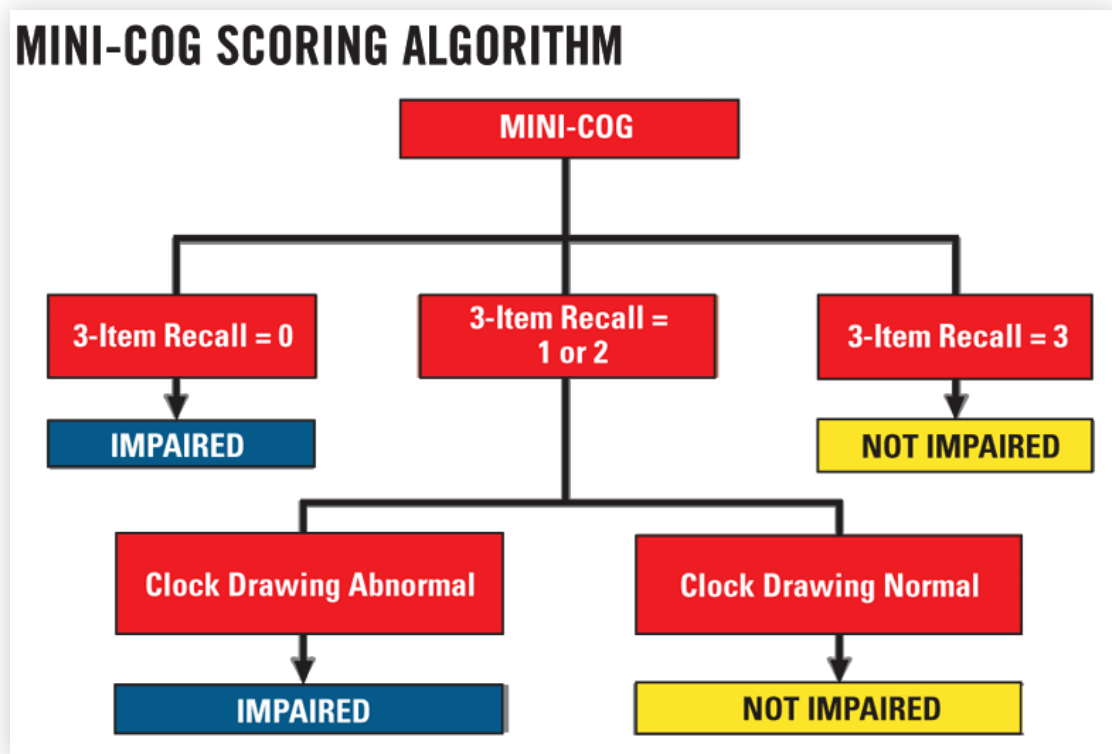


Figure 2.4 – Mini-Cog Scoring Algorithm

2.2.3 MoCa Test

Montreal Cognitive Assessment (MoCA) was published in 2005 by a group at McGill University working for several years at memory clinics in Montreal. It's a test that takes around 25 minutes to complete. It assesses different types of cognitive abilities, including orientation, short-term memory, executive function, language abilities, and visuospatial ability. Unlike the Mini-Mental State Exam (MMSE), another widely used method of screening for Alzheimer's disease, the MoCA includes a clock-drawing test and a test of executive function known as Trails B.

Scores on the MoCA range from 0 to 30, with a score of 26 and higher generally considered normal. In the initial study data establishing the MoCA, normal controls had an average score of 27.4, compared with 22.1 in people with Mild Cognitive Impairment (MCI), and 16.2 in people with Alzheimer's disease.

MOCA SCORES			
	Normal Controls (NC)	Mild Cognitive Impairment (MCI)	Alzheimer's Disease (AD)
Number of subjects	90	94	93
MoCA average score	27.4	22.1	16.2
MoCA standard deviation	2.2	3.1	4.8
MoCA score range	25.2 – 29.6	19.0 – 25.2	21.0 – 11.4
Suggested cut-off score	≥26	<26	<26 ^ψ
^ψ Although the average MoCA score for the AD group is much lower than the MCI group, there is overlap between them. The suggested MoCA cut-off score is thus the same for both. The distinction between AD and MCI is mostly dependent on the presence of associated functional impairment and not on a specific score on the MoCA test.			

Figure 2.5 – MoCa Scores

The MoCA helps health professionals determine quickly whether a person has abnormal cognitive function and may need a more thorough diagnostic work-up for Alzheimer's disease. It may predict dementia in people with Mild Cognitive Impairment (MCI), and because it tests for executive function it is useful in people with scores of 26 or higher on the MMSE. Finally, it has been shown to identify cognitive problems in people with Parkinson's disease.

Advantages of MoCa include brevity, simplicity, and reliability as a screening test for Alzheimer's disease. In addition, it measures an important component of dementia that's not measured by the MMSE, namely executive function. It seems to work well in Parkinson's disease Dementia, and unlike the MMSE it is free for nonprofit use. A disadvantage of the MoCA is that conclusions regarding its validity can only be made in memory clinic settings.

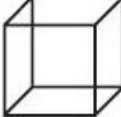
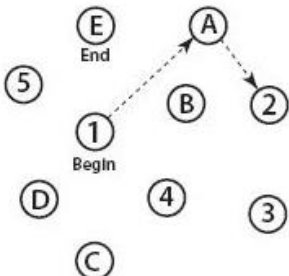
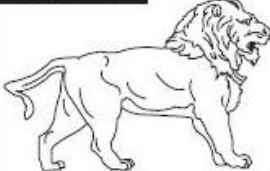
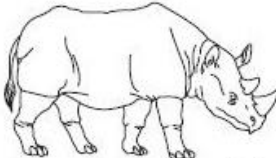
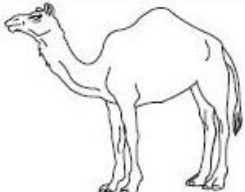
MONTREAL COGNITIVE ASSESSMENT (MOCA)				NAME : _____		Education : _____		Date of birth : _____																				
				Sex : _____		DATE : _____																						
VISUOSPATIAL / EXECUTIVE				 Copy cube		Draw CLOCK (Ten past eleven) (3 points)				POINTS																		
				<input type="checkbox"/> <input type="checkbox"/>		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>				___/5																		
NAMING										___/3																		
MEMORY				Read list of words, subject must repeat them. Do 2 trials, even if 1st trial is successful. Do a recall after 5 minutes.		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td></td> <td>FACE</td> <td>VELVET</td> <td>CHURCH</td> <td>DAISY</td> <td>RED</td> </tr> <tr> <td>1st trial</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>2nd trial</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>					FACE	VELVET	CHURCH	DAISY	RED	1st trial						2nd trial						No points
	FACE	VELVET	CHURCH	DAISY	RED																							
1st trial																												
2nd trial																												
ATTENTION				Read list of digits (1 digit/ sec).		Subject has to repeat them in the forward order <input type="checkbox"/> 2 1 8 5 4 Subject has to repeat them in the backward order <input type="checkbox"/> 7 4 2				___/2																		
Read list of letters. The subject must tap with his hand at each letter A. No points if ≥ 2 errors				<input type="checkbox"/> FBACMNAAJ KLBFAKDEAAA JAM OFAAB		___/1																						
Serial 7 subtraction starting at 100 <input type="checkbox"/> 93 <input type="checkbox"/> 86 <input type="checkbox"/> 79 <input type="checkbox"/> 72 <input type="checkbox"/> 65				4 or 5 correct subtractions: 3 pts, 2 or 3 correct: 2 pts, 1 correct: 1 pt, 0 correct: 0 pt		___/3																						
LANGUAGE				Repeat : I only know that John is the one to help today. <input type="checkbox"/>		The cat always hid under the couch when dogs were in the room. <input type="checkbox"/>				___/2																		
Fluency / Name maximum number of words in one minute that begin with the letter F <input type="checkbox"/> _____ (N ≥ 11 words)				<input type="checkbox"/> _____ (N ≥ 11 words)		___/1																						
ABSTRACTION				Similarity between e.g. banana - orange = fruit <input type="checkbox"/> train - bicycle <input type="checkbox"/> watch - ruler		___/2																						
DELAYED RECALL				Has to recall words WITH NO CUE		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>FACE</td> <td>VELVET</td> <td>CHURCH</td> <td>DAISY</td> <td>RED</td> </tr> <tr> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> </table>				FACE	VELVET	CHURCH	DAISY	RED	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Points for UNCUE recall only								
FACE	VELVET	CHURCH	DAISY	RED																								
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																								
Optional				Category cue		<input type="checkbox"/>				___/5																		
				Multiple choice cue		<input type="checkbox"/>																						
ORIENTATION				<input type="checkbox"/> Date <input type="checkbox"/> Month <input type="checkbox"/> Year <input type="checkbox"/> Day <input type="checkbox"/> Place <input type="checkbox"/> City		___/6																						
© Z.Nasreddine MD Version 7.1 www.mocatest.org				Normal $\geq 26 / 30$		TOTAL ___/30																						
Administered by: _____				Add 1 point if ≤ 12 yredu																								

Figure 2.6 – MoCa Test Sample

Chapter 3 – Technology part

3.1 Mobile applications approach

3.1.1 Native applications

Native applications are, as the name suggests, “real” programs that run directly on a device at the operating system level. They therefore need to be installed on a smartphone or tablet. Installation requires a kind of effort – the user has to visit an app store such as iTunes or Google Play (in the case of Android, the apps can also be downloaded and installed without a store).

A native app is based on a binary code which, once started, interacts directly with the underlying mobile operating system, such as iOS or Android. All APIs that a mobile system and its hardware have to offer can therefore be accessed. This generally gives the developer more options and easier access to the integrated sensors such as gyroscope and positioning, and much more. Furthermore, the special features of a system can also be used more intensively.

For developing a native application, source codes must be written in a language appropriate for the specific OS such as Objective-C and C++ for iOS, Java for Android and Blackberry OS, Visual Basic and C# for Windows Phone. Also multimedia content such as images and audio files must be supplied. The apps are developed in an environment such as Xcode for iOS programs, Android SDK or Visual Studio for Windows programs. These environments are available by the provider of the mobile operating system.

	Apple iOS	Android	Blackberry OS	Windows Phone
Languages	Objective-C, C, C++	Java (some C, C++)	Java	C#, VB.NET and more
Tools	Xcode	Android SDK	BB Java Eclipse Plug-in	Visual Studio, Windows Phone development tools
Packaging format	.app	.apk	.cod	.xap
App stores	Apple App Store	Google Play	Blackberry App World	Windows Phone Marketplace

Figure 3.1 – Features overview of Native Apps Programming

The major disadvantage of native development is that the code written for a mobile platform cannot simply be tailored to another platform.

Maintaining native apps also requires a lot of effort. Security updates, bug fixes and enhancements, etc. cannot simply be loaded onto the server in a matter of minutes as with a web application (described below). Instead, the operators have to communicate the updates to the relevant application stores and the users need to install the updates.

The process for a user to obtain a native application is well defined- users will know exactly where to go to download the app, due to the fact native applications are submitted to stores, like Google Play or the Apple Store. This means also means that a native application must first go through an approval process before it can be made available for download, which adds time and friction.

Summing up:

Pros:

- Lots of design options
- Optimal utilization of hardware and operating system

Cons:

- Complex development
- Complex maintenance
- Difficult multi-platform tailoring

Some Native Tools:

- XCode

- Android SDK
- Eclipse
- Android Studio
- Visual Studio

Another important set of APIs that the OS provides is the GUI toolkit. Each mobile OS comes with its own set of user interface components, such as buttons, input fields, sliders, menus, tab bars, dialog boxes and so on. Apps that make use of these components inherit the features and functions of that specific mobile OS, which normally results in a very easy and enjoyable user experience.

It's important to note that different mobile platforms carry unique palettes of user interface (UI) components. As a result, applications that are designed to work for multiple operating systems require from the developer to use the different UI components of each OS.

Although APIs are OS-specific add much complexity and cost to the development of multiple native apps, these elements are the only means of creating rich mobile applications that make full use of all the functionality that modern mobile devices have to offer.

3.1.2 Web applications

Mobile devices are equipped with web browsers which, with HTML5, CSS3 and JavaScript offer a big variety of possibilities such as Animated, interactive control elements, integration of videos and sounds, positioning functions and offline operation. These technologies allow developing websites and furthermore real software applications that may run in web browsers.

The mobile application may be implemented by web developers; unlike native apps no special expertise is required. Application updates are also performed as usual, we simply install them on the server and we are finished. This method gives users up-to-date data straight away, without them having to maintain and install an update from an app store as in native application. As a conclusion web application are particularly the best choice for all scenarios that require application to be extremely up to date and easy to maintain.

Web applications look like websites but they are able to provide touchscreen-oriented operation, for example with large buttons and swipe gestures. It is acceptable to tell that some even go so far in terms of appearance and operation that they can hardly be distinguished from a native app. In case the start screen holds the links looking like a symbol of a native app, the only difference is that the browser is still present as the environment.

However, all these advantages come at a price – every web app, even if it is disguised well, always runs as part of an external native app, the web browser of the mobile device. For this reason, there is only limited access to APIs. Many functions of the hardware can be accessed only partially, slowly or not at all so as a conclusion certain applications ideas using hardware can only be developed through native application.

Summing up:

Pros:

- Existing expertise in web technologies can be used

- Cheap development
- Fast and frequent updates easily possible
- Wide range of functions possible thanks to HTML 5

Cons:

- Always runs in the browser environment
- Often less convenient than a native app
- Limited offline operation

Feature	Pure mobile web apps	Pure mobile websites
Tools and knowledge	Written entirely in HTML, CSS and JavaScript	Written entirely in HTML, CSS and JavaScript
Execution	"Installed" shortcut, launched like a native app	Reached by navigating to a website by way of a Uniform Resource Locator (URL)
User experience	Touch-friendly, interactive UI	Navigational UI between pages displaying static data
Performance	UI logic resides locally, making the app responsive and accessible offline	All code executed from a server, resulting in network-dependent performance

Figure 3.2 – Features overview of Web Apps Programming

Device APIs for Web Apps:

- Apache Cordova
- Qualcomm HTML5 Device APIs
- APIs from HTML5 Rocks
- Motion data (accelerometer & gyroscope) from Mozilla
- Camera API from Mozilla
- Device Orientation API
- getUserMedia API
- Airplay API
- Speech Synthesis API
- New APIs in Mobile Safari

3.1.3 Hybrid applications

It's only natural that from the native app and web app, each with their own set of advantages and disadvantages, would come a third classification of apps – the hybrid app. Hybrid applications are native apps that leverage “web views” to display HTML content and allows us to get the best of both web apps and native apps. Web views are native widgets that display web pages but lack the address bar and navigation controls that appear in a normal browser. This lets developers embed web content inside an app and seamlessly blend both native elements and web content on the same screen.

With the hybrid approach, developers can program a large part of the application using web technologies, while access is also available to native APIs. This compromise approach uses a trick – the native part of the app runs directly on the operating system. It provides an HTML rendering engine and a kind of “bridge” to the operating system. This way, a web code can run in its environment and, thanks to the connection via the bridge, still use all features of the hardware and system.

Again, there are many options – in theory, developers can program their own bridge. Alternatively, they can access ready-made solutions. An example of this is PhoneGap, an open source library based on JavaScript.

The HTML component can be hosted on a server like a web app, which makes minor updates possible without having to update the entire app using the complex update process. However, this approach does not work offline, as the HTML content is only available when connected to the Internet. It can also be supplied in the app as a package for offline use, which improves performance; however, online updates are then not an option again. There is, therefore, always a drawback. A compromise can be reached by hosting the HTML resources on the website for flexibility, but regularly saving them locally on the mobile device.

Summing up:

Pros:

- Combined advantages of the web app and native development approaches
- Bridges between the native app and web code

Cons:

- Impaired performance (when accessing web content online)
- No frequent updates (in the case of offline caching)

While web and native app classifications are binary, the hybrid app is much more of a spectrum. It's entirely possible - and quite common - to develop an app that is almost entirely built from HTML, CSS, and Javascript, only to add later native elements like the static bottom menu for navigation, GPS calls for location lookup, or camera access for barcode scanning. Also it is possible to develop a mostly native app with only a few screens powered by web views.

Hybrid App Tools:

- PhoneGap (Apache Cordova) is a free and open source framework that allows you to create mobile apps using standardized web APIs.
- Trigger.io is a mobile development platform that bundles a hybrid framework with a JavaScript API for device functionality and UI components with a cloud based build environment.
- IBM Worklight is an application development platform that extends PhoneGap (Apache Cordova) with additional development tools and server support for backend integration, authentication, push notifications, and life cycle management.
- Sencha Space is a platform for securely distributing HTML5 and hybrid business applications in a managed enterprise environment.

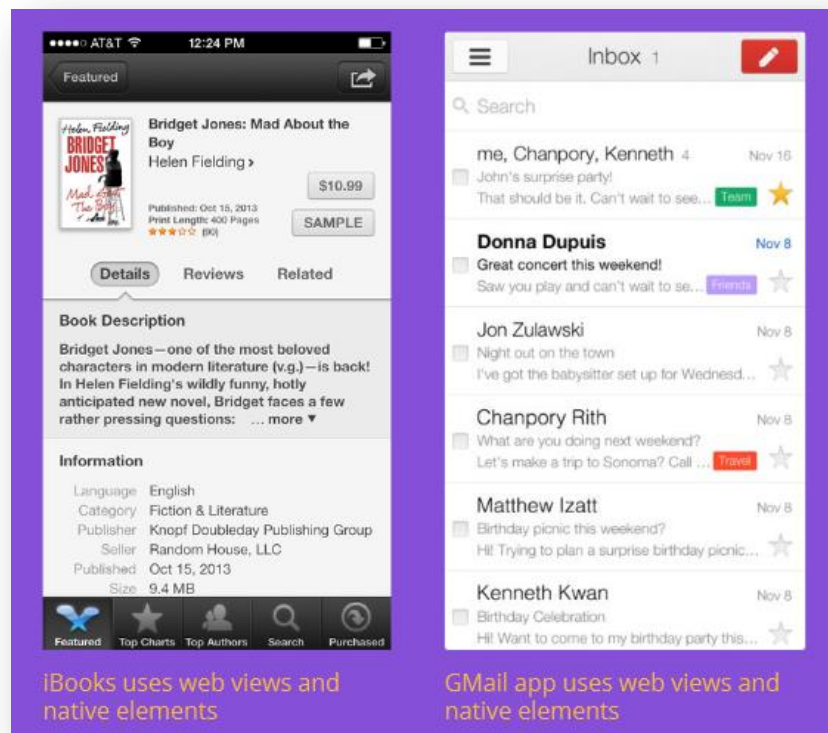


Figure 3.3 – Examples of Hybrid Apps

3.1.4 Comparing different approaches

The native approach excels in performance and device access, but suffers in cost and updates. The web approach is much simpler, less expensive and easier to update, but is currently limited in functionality and cannot achieve the exceptional level of user experience that can be obtained using native API calls. The hybrid approach provides a middle ground which, in many situations, is the best of both worlds, especially if the developer is targeting multiple operating systems.

Feature	Native app	Hybrid app	Web app
Development language	Native only	Native and web or web only	Web only
Code portability and optimization	None	High	High
Access device-specific features	High	Medium	Low
Leverage existing knowledge	Low	High	High
Advanced graphics	High	Medium	Medium
Upgrade flexibility	Low (Always by way of app stores)	Medium (Usually by way of app stores)	High
Installation experience	High (From app store)	High (From app store)	Medium (By way of mobile browser)

Figure 3.4 – Comparison between mobile application approaches



Figure 3.5 – Schematic presentation of mobile apps

3.2 Development App tools

3.2.1. PhoneGap

PhoneGap is a mobile development framework around 2009 by a startup called Nitobi as an open source way to access the "native" environment through an embedded Web View in a native app. The goal of the project was to make it possible to build the bulk of a mobile app experience with pure web technologies like HTML5, CSS3, and Javascript, but still be able to call into native code when necessary.

The core of PhoneGap applications uses HTML5 and CSS3 for their rendering and JavaScript for their logic. PhoneGap framework embeds HTML5 code inside a native WebView on the device, using a foreign function interface to access the native resources of the device.

PhoneGap is also able to be extended with native plug-ins that allow for developers to add functionality that can be called from JavaScript, allowing direct communication between the native layer, and the HTML5 page..

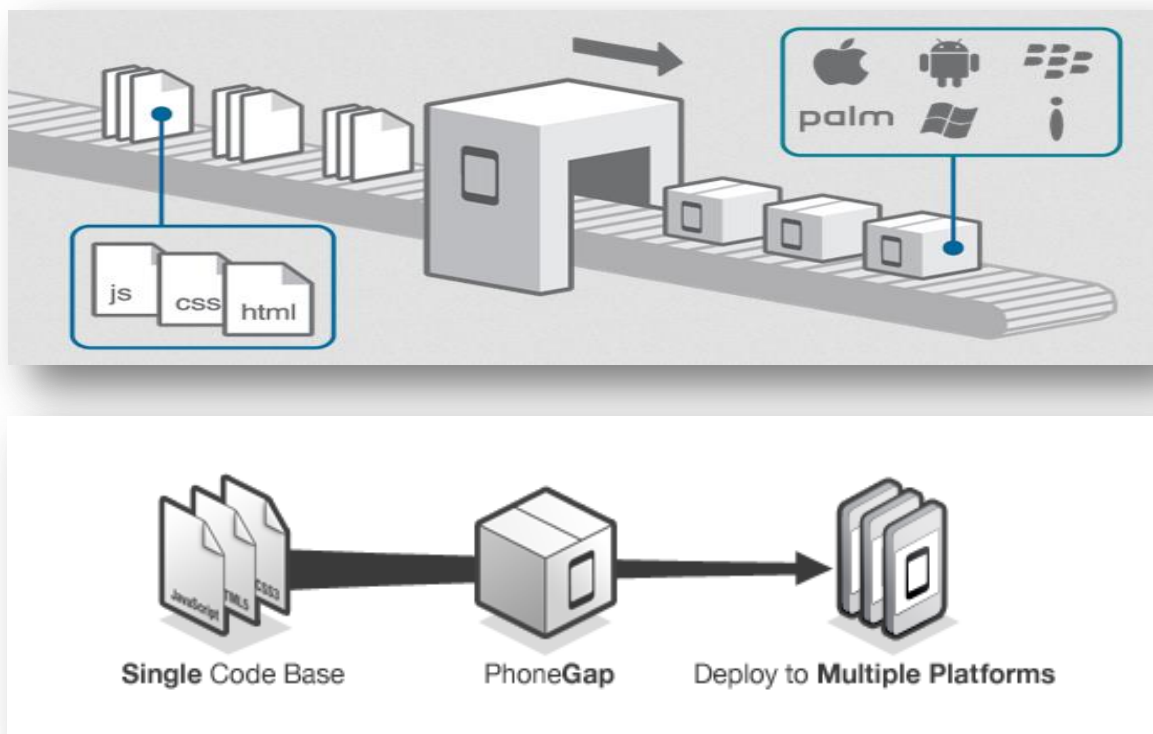


Figure 3.6 – PhoneGap Schematic Overview

In 2011 Adobe purchased Nitobi and with it the rights to the PhoneGap brand, and the open source core was donated to the Apache Software Foundation under the name Cordova.

3.2.2 Cordova

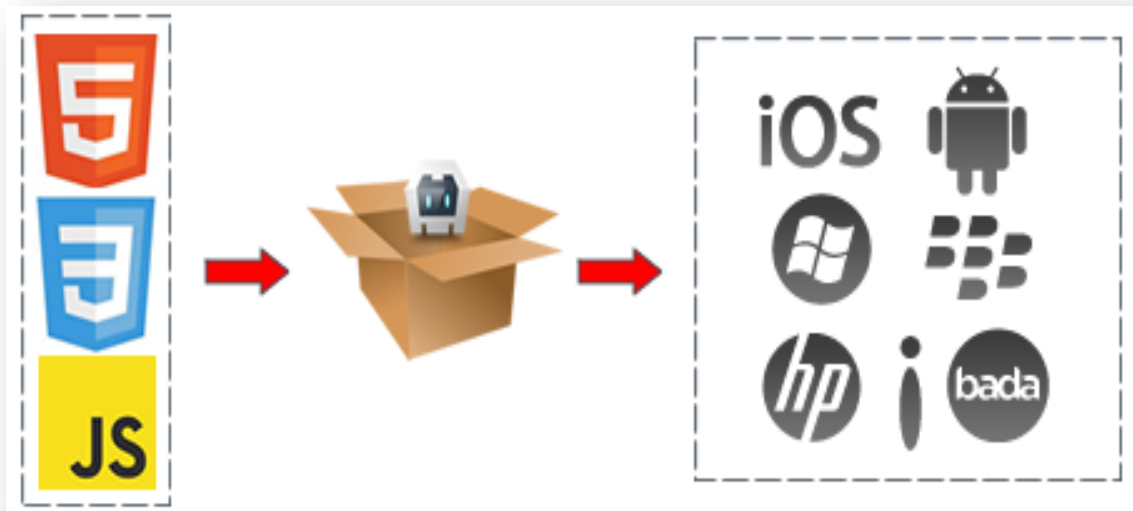


Figure 3.7 – Cordova Schematic overview

When using the Cordova APIs, a mobile application can be built without any native code (Java, Objective-C, etc) from the developer. Instead, web technologies are used, and they are hosted in the app itself mostly locally and rarely on a remote http server.

Using these APIs the application is portable to device platforms with minimal changes or not at all.

Cordova is available for the following platforms: iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, and Symbian.

Apps using Cordova are still packaged as apps using the platform SDKs, and can be made available for installation from each device's app store.

Cordova provides a set of uniform JavaScript libraries that can be invoked, with device-specific native backing code for those JavaScript libraries.

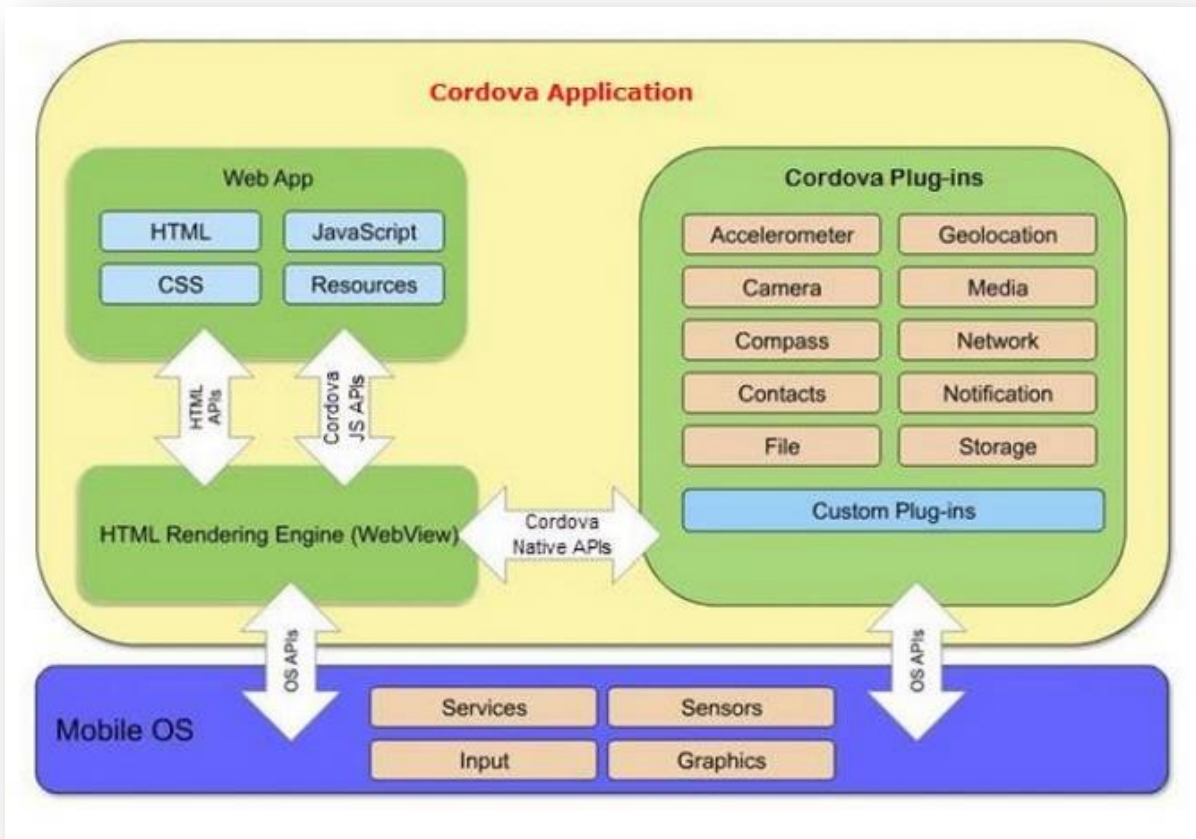


Figure 3.8 – Cordova Architecture

- * The user interface for Apache Cordova applications is created using HTML, CSS, and JavaScript.
- * The UI layer is a web browser view that takes up 100% of the device width and 100% of the device height.
- * The web view used by application is the same web view used by the native operating system
 - * iOS: **Objective-C UIWebView class**
 - * Android: **android.webkit.WebView**
 - * WP7: **WebBrowser**
 - * WP8: **WebBrowser control (Internet Explorer 10)**
 - * BlackBerry: **WebWorks framework**



Figure 3.9 – Cordova UI

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7 + 8	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	X	✓	✓	X	✓
Contacts	✓	✓	✓	✓	✓	X	✓	✓	✓
File	✓	✓	✓	✓	✓	X	✓	X	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	X	X	✓	X	X
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	X	X

Figure 3.10 – Supported Features byCordova

3.2.3 Eclipse

Eclipse is a multi- language software development environment containing an IDE and a plug-insystem. Eclipse is written in Java and the IDE is a particular favourite with Java developers. The IDE can also be used to write applications in other programming languages, i.e. Ada, C and C++. The Eclipse SDK is meant for Java developers and users can extend its capabilities by installing plugins for the Eclipse platform. For Android development in Eclipse the plug in Android Development Tools (ADT) is installed.

Eclipse SDK is free and open source software under the terms of Eclipse Public License.

3.3 Programming Languages

3.3.1 HTML5

The HTML5 standard began to be written in 2004 by the Web Hypertext Application Technology Working Group (WHATWG), while the World Wide Web Consortium (W3C) was working on XHTML 2.0. This standard was born from the disagreement between browser vendors and the W3C on how the Web could be fixed.

HTML5 is the evolution of HTML4, a standard that adds many new functionalities and APIs to web browsers, trying to supersede external plugins like Flash, Silverlight and Java. These include audio and video support, scalable vector graphics, drag-and-drop, history management, web storage, geo-location and many others.

Enabling sites to store data larger than a cookie in the client's web browser is also a functionality included in the standard. It enables the developer to cache files, images or data manually. HTML5 also provide offline caching, where the resources of the web site can be cached and used for running it when no connectivity exists.

HTML5 is indeed bringing many new features to web browsers, including the mobile ones. The standard is not yet finished, but when it does and the most common browsers implement it completely, web applications based on it will truly be cross-platform. At the moment, most browsers already implement most parts of the standard, especially the mobile browsers of Android and iOS.

Each mobile operating system provides its own software development kit. For instance, they differ in programming languages: Java is used to build applications for Android while Objective-C must be used for developing applications for iOS. Other platforms also use other programming languages. This means that an application developed for one platform, cannot be used in another without the code being ported.

This is where HTML5 has the upper hand. Using HTML5, CSS3 and Javascript, developers can target several platforms were the code once.

There are already many frameworks and libraries that enable developers to use these technologies to build mobile applications, as is shown in the next section.

HTML5 allows for a single codebase which developers can deploy on multiple platforms quickly; they can write it once and have that same version run on several operating systems. Platform-specific teams aren't necessary, and the burden of building one app over and over to run on multiple devices is lifted. This is particularly beneficial to organizations looking to build apps for a variety of departments and tasks.

HTML5 allows developers to bypass the approval process of app stores. Giving updates directly allows developers to bypass the approval process of app stores. Changes are simply pushed to the production servers, and users see the updates immediately. In previous versions of iOS, for example, users had to manually download app updates, giving HTML5 another leg up on that front. However, iOS 7 introduced automatic app updates, making for a more seamless update experience similar to HTML5. The real advantage of web apps here is forgoing any sort of approval process from app stores, therefore getting the improved app into the hands of users faster.

3.3.2 CSS3

Cascading Style Sheets (CSS) is a kind of language for web pages that enables elements such as layout, colours and fonts to be personalized, improving accessibility and flexibility in the formatting, as well as a reduction in the complexity and repetition in the contents. As part of the W3C web standards, most browsers today support CSS3, and this includes all major desktop browsers (Chrome, Firefox, Internet Explorer, Safari and Opera), as well as WebKit for mobile browsers.

Although desktop browsers fully support both CSS3 and Adobe Flash, there is a known incompatibility between the later and iPhone mobile devices, and Adobe has recently announced that they won't be developing Flash for mobile devices any more. Therefore, CSS3, in combination with HTML5, has arisen as an alternative of Flash for mobile devices, providing videos, graphics and animations for web pages.

Applied to smartphones, HTML5 in combination with CSS3 is very useful, because it allows the creation of user interfaces similar to native apps, without the restrictions of each platform and without the need to multiply the versions of the app for each one of them. This way, apps are easier to develop and they can be more easily cross-platform spread. Combine them is the future of web design and can be used today.

3.3.3 AngularJS

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as template language and extend HTML's syntax to express your application's components clearly and succinctly.

Angular is what HTML would be if it was designed for applications. Now HTML is a great declarative language for static documents.

The impedance mismatch between dynamic applications and static documents is often solved with:

- a library - a collection of functions which are useful when writing web apps. In simple talking code in in charge and it calls into the library when it sees fit like jQuery.
- frameworks - a particular implementation of a web application, where code fills in the details. The framework is in charge and it calls into code when it needs something app specific.as durandal

Angular takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. It is important to mention that Angular teaches the browser new syntax through a construct we call directives.

Angular is used in order to avoid:

Registering callbacks: It vastly reduces the amount of JavaScript coding you have to do, and it makes it easier to see what your application does.

Manipulating HTML DOM programmatically: is a cornerstone of AJAX applications, but it's cumbersome and error-prone. By declaratively describing how the UI should change as your application state changes, we are freed from low-level DOM manipulation tasks.

Marshaling data to and from the UI: CRUD operations make up the majority of AJAX applications' tasks. The flow of marshaling data from the server to an internal object to an HTML form, allowing users to modify the form, validating the form, displaying validation errors, returning to an internal

model, and then back to the server, creates a lot of boilerplate code. Angular eliminates almost all of this boilerplate, leaving code that describes the overall flow of the application rather than all of the implementation details.

Writing tons of initialization code just to get started: With Angular we can bootstrap our app easily using services, which are auto-injected into your application in a Guice-like dependency-injection style. This allows you to get started developing features quickly. As a bonus, you get full control over the initialization process in automated tests. Angular initializes automatically upon DOMContentLoaded event or when the angular.js script is evaluated if at that time document.readyState is set to 'complete'. At this point Angular looks for the ng-app directive which designates your application root. If the ng-app directive is found then Angular will:

- load the module associated with the directive.
- create the application injector
- compile the DOM treating the ng-app directive as the root of the compilation. This allows you to tell it to treat only a portion of the DOM as an Angular application.

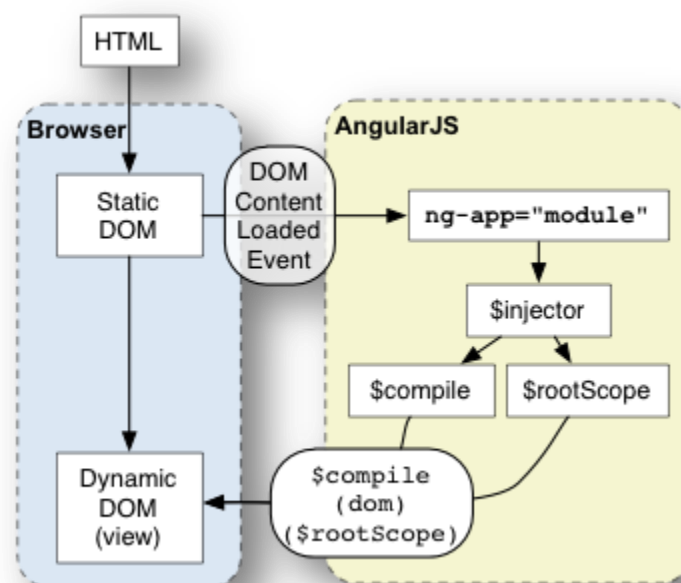


Figure 3.11 – Initialization of AngularJs

If order to gain more control over the initialization process, we may use a manual bootstrapping method instead. Due to the fact angular.bootstrap cannot create modules on the fly we first create any custom modules and then pass them as a parameter. We should call angular.bootstrap() after

modules are defined. We add controllers, services, directives, etc before the application bootstraps.

Modules/Libraries

AngularLocalStorage Build Status - The simplest localStorage module that allows setting, getting, and binding variables. Features: Two way of bind \$scope variable value to a localStorage key/pair which will be updated whenever the model is updated.

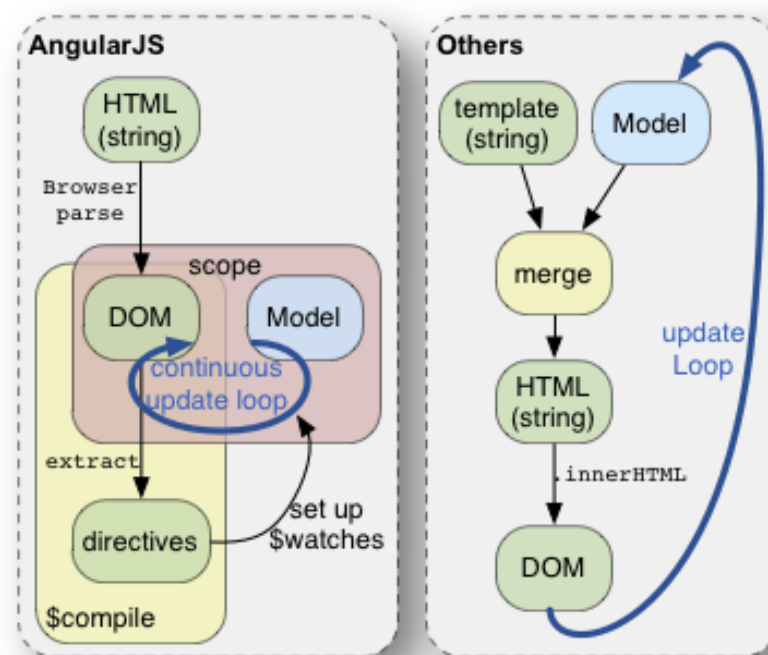


Figure 3.12 – AngularJs Update Schema

Moment.js - library to Parse, validate, manipulate, and display dates in javascript

Sugar.js - library that extends native objects with helpful methods. It is designed to be intuitive, unobtrusive, and let you do more with less code.

angular-loader.min.js - Module loader for Angular modules. For loading multiple script files containing Angular modules, we load them asynchronously and in any order. Often the contents of this file are copy&pasted into the index.html to avoid even the initial request to angular-loader.min.js.

angular-animate.js - Enable animation support

angular-cookies.js - A convenient wrapper for reading and writing browser cookies

angular-resource.js - Interaction support with RESTful services via the \$resource service

angular-route.js - Routing and deeplinking services and directives for angular apps

angular-sanitize.js - Functionality to sanitize HTML

angular-touch.js - Touch events and other helpers for touch-enabled devices such as tablets and smartphone

Directives

At a high level, directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler (\$compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children. For AngularJS, "compilation" means attaching event listeners to the HTML to make it interactive.

Angular comes with a set of these directives built-in, like ngBind, ngModel, and ngView. When Angular bootstraps the application, the HTML compiler traverses the DOM matching directives against the DOM elements.

Bootstrap is made with LESS at its core, a dynamic stylesheet language. It makes developing systems-based CSS faster and easier. Bootstrap compiles faster ~6x faster with Less. Less is written in JavaScript, making it easier to dive. When we use the source .less files instead of our compiled CSS files we can make use of what we use throughout the framework. Variables are used throughout the entire project as a way to centralize and share commonly used values like colors, spacing, or font stacks.

Services

Angular services are substitutable objects that are wired together using dependency injection (DI). Services are used to organize and share code across the application.

- ✓ Lazily instantiated – Angular only instantiates a service when an application component depends on it.
- ✓ Singletons – Each component dependent on a service gets a reference to the single instance generated by the service factory.

Models

A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of Model View Controller (MVC) omits these notifications, because the application does not require them or the software platform does not support them.

JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

Although originally derived from the JavaScript scripting language, JSON is a language-independent data format, and code for parsing and generating JSON data is readily available in a large variety of programming languages.

Filters

A filter formats the value of an expression for display to the user. They can be used in view templates, controllers or services and they are mostly designed by the programmer.

Custom Form Validation

Angular provides basic implementation for most common HTML5 input types: (text, number, url, email, radio, checkbox), as well as some directives for validation (required, pattern, minlength, maxlength, min, max).

Defining our validator can be done by defining directive which adds a custom validation function to controller. The validation can occur in two places:

- ✓ Model to View update - Whenever the bound model changes, all functions in array are pipe-lined, so that each of these functions has an opportunity to format the value and change validity state of the form control
- ✓ View to Model update - In a similar way, this in turn pipelines all functions in the array, so that each of these functions has an opportunity to convert the value and change validity state of the form control.

Controllers

In Angular, a Controller is a JavaScript constructor function that is used to augment the Angular Scope.

When a Controller is attached to the DOM via the ng-controller directive, Angular will instantiate a new Controller object, using the specified Controller's constructor function. A new child scope will be available as an injectable parameter to the Controller's constructor function as \$scope.

We use controllers to:

- Set up the initial state of the \$scope object.
- Add behavior to the \$scope object.

Data Binding Angular Templates

First the template (which is the uncompiled HTML along with any additional markup or directives) is compiled on the browser. The compilation step produces a live view. Any changes to the view are immediately reflected in the model, and any changes in the model are propagated to the view. The model is the single-source-of-truth for the

application state, greatly simplifying the programming model for the developer.

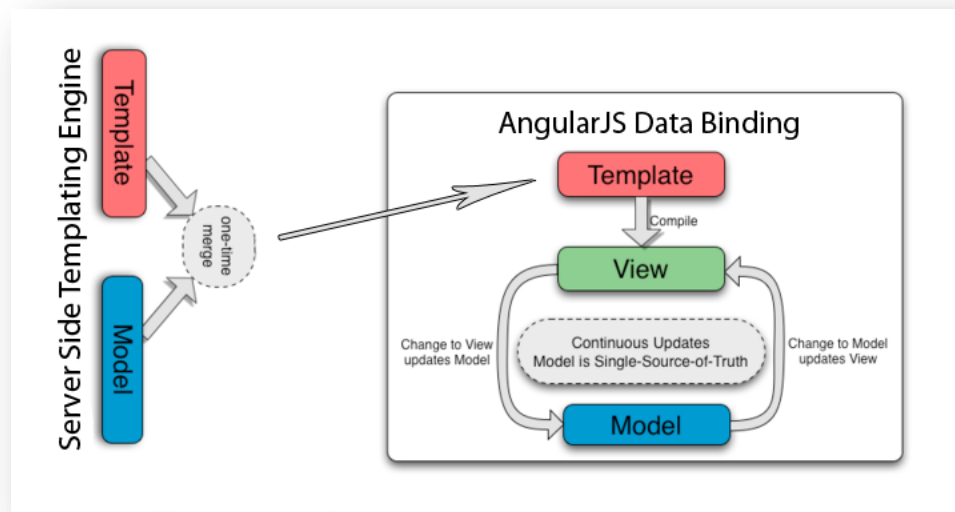


Figure 3.13 – AngularJS Data Binding

3.4 Application Setup & Technical Analysis

In order install Cordova we needed the following prerequisites:

- node.js – this includes the npm package manager, and is the gateway to a world of open source tools and packages
- Java JDK - We need to download the Java Development Kit (JDK), not just the JRE that does not have development tools.
- Apache ANT – A scripting framework
- Android Developer SDK

Now we are able to install Cordova using the package manager we installed before with node.js.

```
$ sudo npm install -g cordova
```

Prefixing the npm command with sudo may be necessary to install this development utility in otherwise restricted directories such as /usr/local/share. If you are using the optional nvm/nave tool or have write access to the install directory, you may be able to omit the sudo prefix.

For Cordova command-line tools to work, or the CLI that is based upon them, you need to include the SDK's tools and platform-tools directories in our PATH.

We may use a text editor to create or modify the ~/.bash_profile file, adding a line such as the following, depending on where the SDK installs:

```
export PATH=${PATH}:/Development/adt-bundle/sdk/platform-  
tools:/Development/adt-bundle/sdk/tools
```

If needed we also must add the paths for java and ant.

In order to create a new project with CLI tool we execute:

```
$ cordova create thesis com.example.thesis ThesisProject  
$ cd thesis  
$ cordova platform add android  
$ cordova build
```

PhoneGap App Architecture Diagram

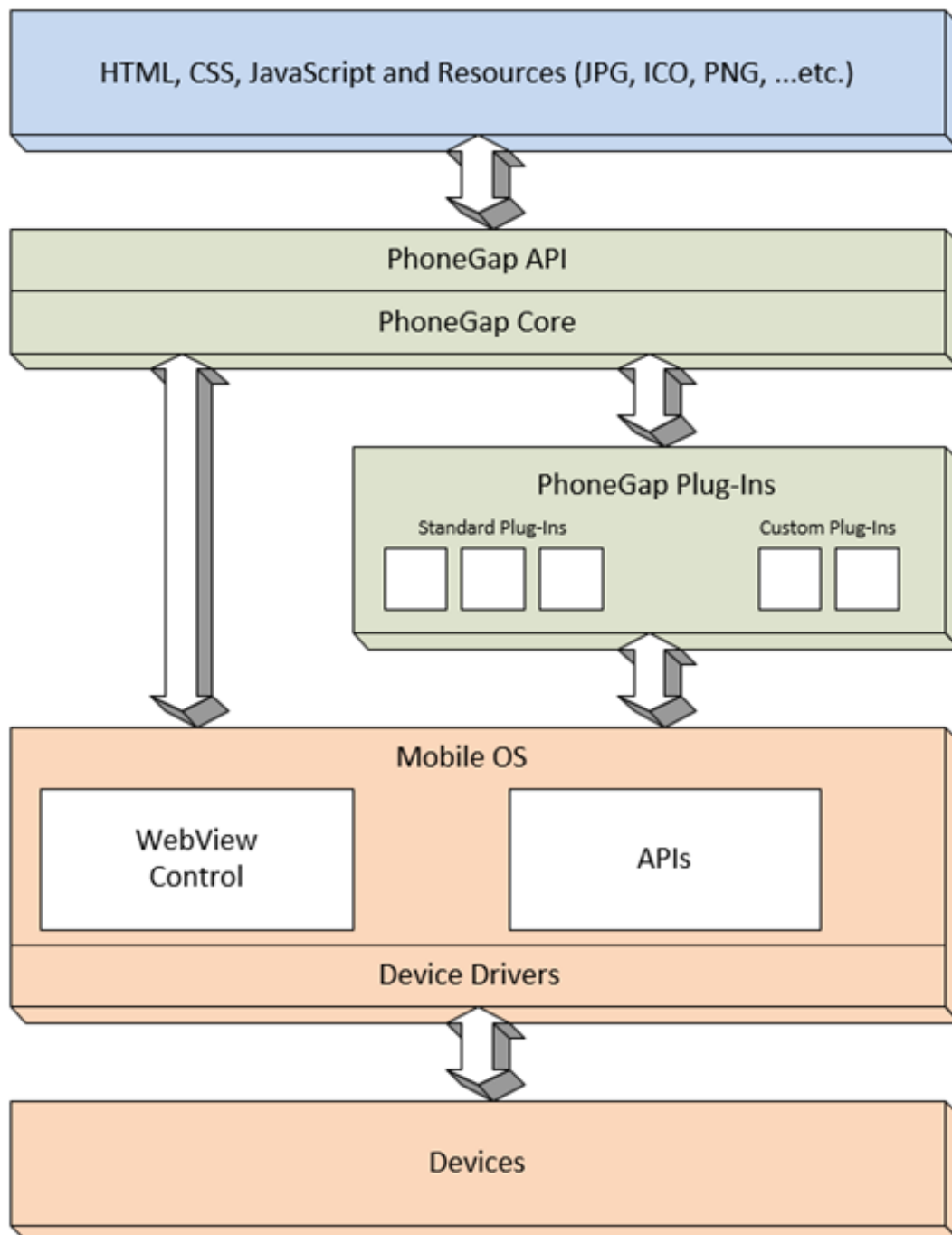


Figure 3.14 – Cordova/ PhoneGap Architecture Diagram

Once we built our project we get the following folder structure made by Cordova:

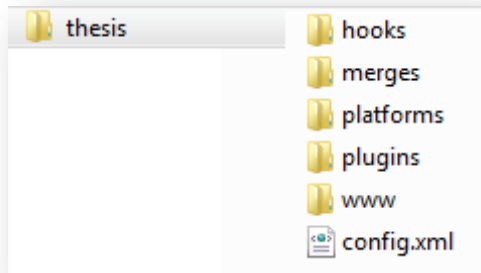


Figure 3.15 – Cordova output folder structure

All modules placed in the appropriate folders provide a way to group dependencies for a functional area of the application and a mechanism for automatically resolving those dependencies as shown below:

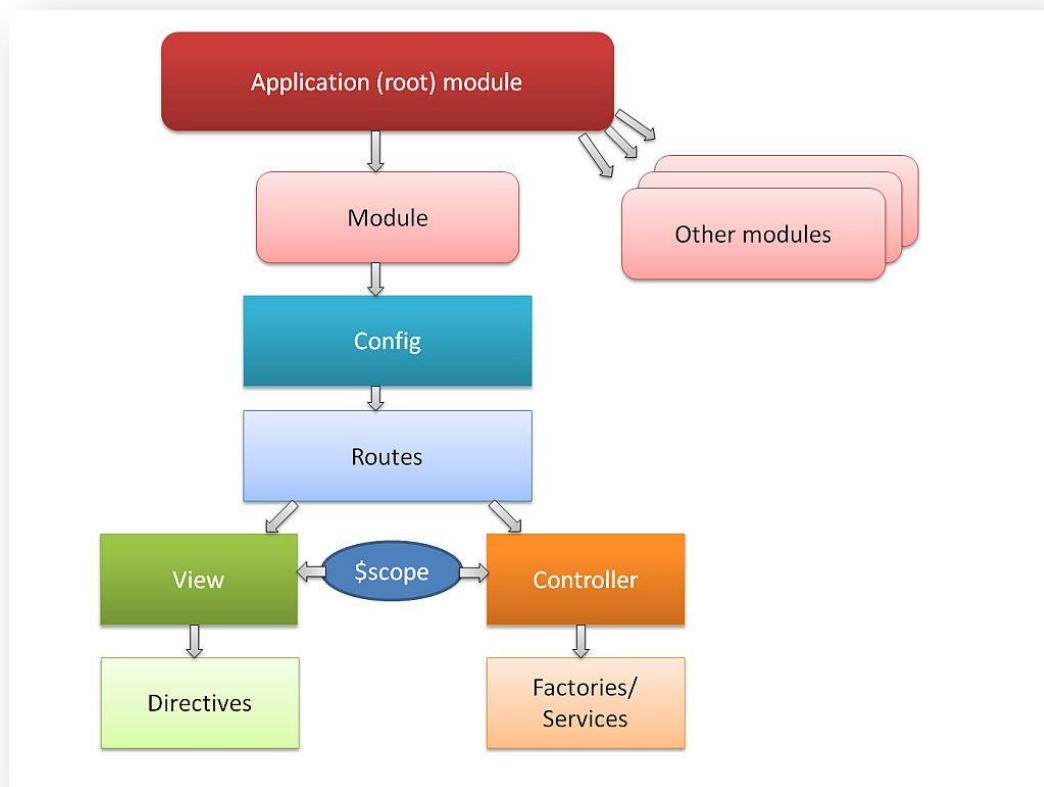


Figure 3.16 – Project flow diagram

In the first place this is implemented by the config.xml file that simply calls index.html under the www folder - where we place code files, libraries, images etc. of our application. The folder structure is:

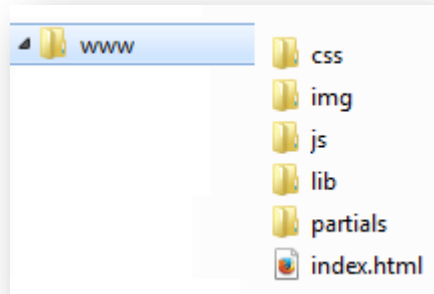


Figure 3.17 – Folder structure of code

In index.html file we include under the ng-view directive, that complements the \$route service, the main app file (app.js), all source files needed (controllers, models, services etc.) and we initialize the app calling app.init().

index.html
<pre> <!doctype html> <html lang="en"> <head> <meta charset="utf-8"> <title>Alzheimer Thesis</title> <link rel="stylesheet" href="css/app.css"/> </head> <body> <div ng-view></div> <script src="cordova.js"></script> <script src="lib/angular/angular.js"></script> <script src="lib/angular/angular-route.js"></script> <script src="lib/angular/angular-cookies.js"></script> <script src="lib/angular/angular-animate.js"></script> <script src="lib/less-1.6.0.min.js"></script> <script src="lib/sugar.min.js"></script> <script src="lib/angularLocalStorage.js"></script> <script src="lib/moment.min.js"></script> <script src="js/app.js"></script> <script src="js/services.js"></script> <script src="js/filters.js"></script> <script src="js/directives.js"></script> <!-- controllers --> <script src="js/controllers/login.js"></script> <script src="js/controllers/history.js"></script> </pre>


```

<script src="js/controllers/dashboard.js"></script>
<script src="js/controllers/clock.js"></script>
<script src="js/controllers/triplet.js"></script>
<script src="js/controllers/pickTriplet.js"></script>
<script src="js/controllers/cube.js"></script>
<script src="js/controllers/sketch.js"></script>
<script src="js/controllers/results.js"></script>
<script src="js/controllers/animals.js"></script>
<script src="js/controllers/words.js"></script>
<script src="js/controllers/categories.js"></script>
<script src="js/controllers/hangman.js"></script>
<script src="js/controllers/subtract.js"></script>
<script src="js/controllers/history.js"></script>

<!-- models -->
<script src="js/models/triplets.js"></script>
<script src="js/models/flowTypes.js"></script>
<script src="js/models/sketches.js"></script>
<script src="js/models/animals.js"></script>
<script src="js/models/words.js"></script>
<script src="js/models/categoriesPairs.js"></script>
<script src="js/models/geometry.js"></script>
<script src="js/models/games.js"></script>

<script>
  app.init();
</script>

</body>
</html>

```

Dependencies are shown in app.js file where we declare application level module and we configure \$routeProvider, in each case which partial is shown and by which controller is managed.

app.js

```

'use strict';

var app = {
  init: function () {
    this.bindEvents();
  },

  bindEvents: function () {
    document.addEventListener("touchstart", function () {
    }, true);
    document.addEventListener('deviceready', this.onDeviceReady, false);
  },

  onDeviceReady: function () {
    console.log('*** device ready');
    angular.bootstrap(document, ['thesis']);
  }
}

// Declare app level module which depends on filters, and services
angular.module('thesis', [

```

```

    'ngRoute',
    'thesis.filters',
    'thesis.services',
    'thesis.directives',
    'angularLocalStorage'
  ]).
  config(['$routeProvider', '$locationProvider', function ($routeProvider, $locationProvider) {
    $routeProvider
      .when('/login', { templateUrl: 'partials/login.html', controller: LoginCtrl })
      .when('/logout', { templateUrl: 'partials/login.html', controller: LoginCtrl })
      .when('/history', { templateUrl: 'partials/history.html', controller: HistoryCtrl })
      .when('/dashboard', { templateUrl: 'partials/dashboard.html', controller: DashboardCtrl })
      .when('/clock', { templateUrl: 'partials/clock.html', controller: ClockCtrl })
      .when('/triplet', { templateUrl: 'partials/triplet.html', controller: TripletCtrl })
      .when('/cube', { templateUrl: 'partials/cube.html', controller: CubeCtrl })
      .when('/pickTriplet', { templateUrl: 'partials/pickTriplet.html', controller: PickTripletCtrl })
      .when('/results', { templateUrl: 'partials/results.html', controller: ResultsCtrl })
      .when('/sketch', { templateUrl: 'partials/sketch.html', controller: SketchCtrl })
      .when('/animals', { templateUrl: 'partials/animals.html', controller: AnimalsCtrl })
      .when('/words', { templateUrl: 'partials/words.html', controller: WordsCtrl })
      .when('/categories', { templateUrl: 'partials/categories.html', controller: CategoriesCtrl })
      .when('/hangman', { templateUrl: 'partials/hangman.html', controller: HangmanCtrl })
      .when('/subtract', { templateUrl: 'partials/subtract.html', controller: SubtractCtrl })
      .otherwise({ redirectTo: '/login' });
  }]);

```

SDKs for mobile platforms often come bundled with emulators that execute a device image, so that you can launch the app from the home screen and see how it interacts with many platform features. We use the following command to rebuild the app and view it within a specific platform's emulator:

```
$ cordova emulate android
```

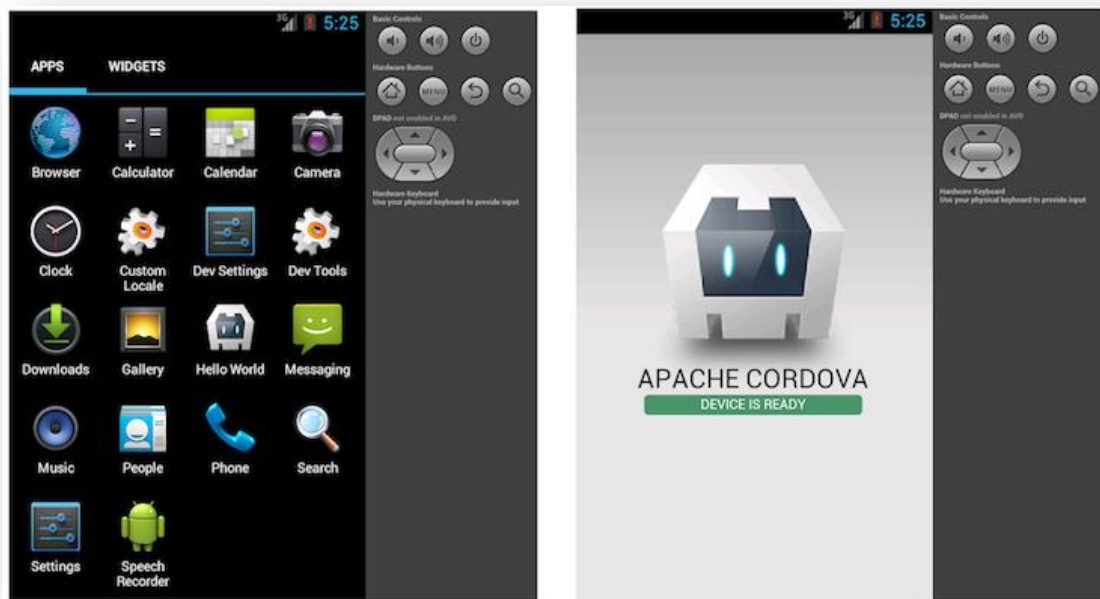


Figure 3.18 – Android emulation

Chapter 4 – Our approach

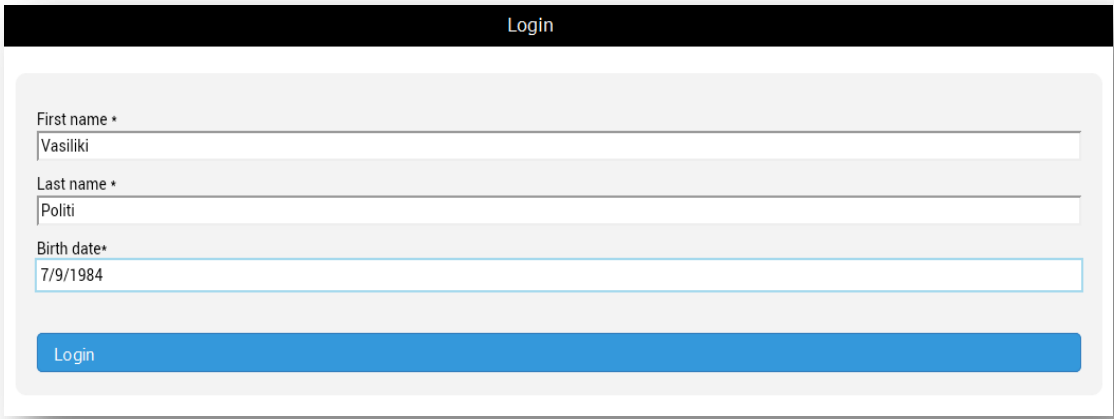
4.1 Methodology

We choose to select Hybrid approach using HTML5, CSS3 and AngularJS combined with Cordova, due to the fact that we are looking to develop the project across a range of devices, but also aiming to take advantage of features that a purely web app cannot give us. Web views enable to reuse code and content that run on the browser for iOS, Android, or Windows apps etc. Only the native elements of a hybrid app need to be rewritten for each new app's operating system. This makes hybrid an ideal choice when multiple platforms need to be supported.

4.2 Tests and Interfaces

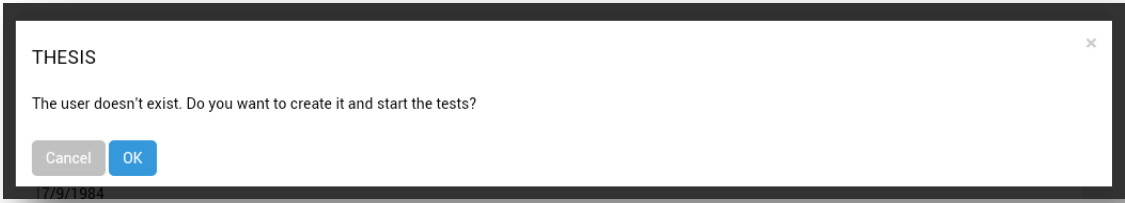
4.2.1 Login screen

As in all clinical studies, the identification information of the patients is saved. Every patient is unique created or recognized by last name and date of birth. An inside application small database holds this kind of information needed. This database also stores all tests the patients gave sorted by date, with analytic scores per test and per category of the test. If the patient is new a prompt appears informing that a new user will be created.



The login screen features a black header bar with the word "Login" in white. Below the header is a light gray form area. It contains three input fields: "First name *" with the value "Vasiliki", "Last name *" with the value "Politi", and "Birth date*" with the value "7/9/1984". At the bottom of the form is a blue button labeled "Login".

Figure 4.1 – Application's Login screen



The prompt screen is a white dialog box with a black border and a close button (X) in the top right corner. The title is "THESIS". The text inside reads: "The user doesn't exist. Do you want to create it and start the tests?". At the bottom are two buttons: "Cancel" (gray) and "OK" (blue). The date "17/9/1984" is visible at the bottom left of the dialog.

Figure 4.2 – Prompt screen for new user

4.2.2 Dashboard

Dashboard is the main screen of the application. Here, we may find all tests implemented, Clock drawing test, Mini Cog Test and MoCa Test.

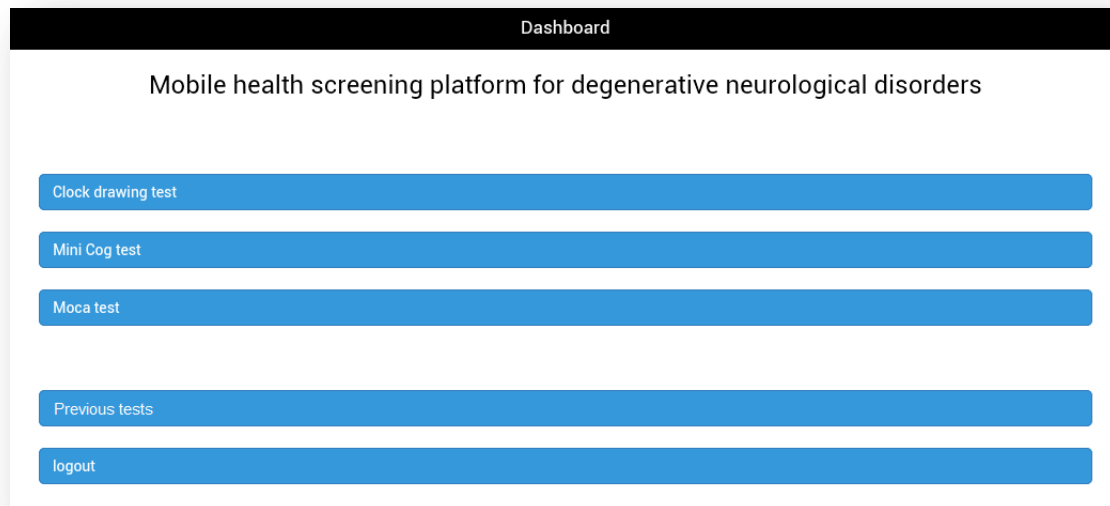


Figure 4.3 – Dashboard

We also find an option for showing the history of previous tests. This option contains statistical information about the score per test that was gained, sorted by date.

4.2.3 Previous tests

In this section as seen in the following picture we may take a look at the tests implemented by one user and the score per test and per category of the test. We also have the opportunity to export these data in CSV format and send them directly via e-mail. This file could then be imported to any application or program or even stored outside the application database.

Tests history		CSV
Vasiliki Politi		
Friday 10 July 2014, 7:29		
Minicog	Total Score:7	
Clock	4	
Triplet	3	
Friday 10 July 2014, 7:43		
Moca	Total Score:11	
Cube	1	
Picture	1	
Clock	4	
Animals	3	
Memory	0	
Attention	1	
Abstraction	3	
Substract	2	

Figure 4.4 – Tests history

4.2.4 Clock drawing test

Goal

The goal is to draw a clock showing time 11:15.

Scoring

Total score is 4.

One point is allocated for each of the following four criteria:

- Contour (1 pt.): the clock face must be a circle (e.g., only slight imperfections are accepted)
- Numbers (2 pt.): all clock numbers must be present with no additional numbers (1pt.) Numbers must be in the correct order and placed in the approximate quadrants on the clock face (1 pt.)
- Hands (1 pt.): there must be two hands jointly indicating the correct time. Hour hand must be clearly shorter than the minute hand.

Screenshots

A blank screen page appears prompting the user to draw a circle that will represent the clock's contour.

In case the user draws something else (usually patients with dementia write their name or draw a line) then another try is given. User has a lot of times to try, but usually the administrator stops the procedure after 5 unsuccessful times.

If the user refuses to do something or cannot draw the circle then the test can be preceded to the next stage using the next button. This can be done also if the user tried unsuccessfully 5 times.

In the next phase an adjusted size of the clock's contour is shown and the user must now fill the correct numbers that represent the hours of a clock. All numbers are included in the drop-down boxes at each place. The fixed contour is made due to the fact many users draw a very big or a very small circle and the numbers were impossible to put.

The numbers showing from drop-down boxes are not sorted just because the user must remember the hours order.

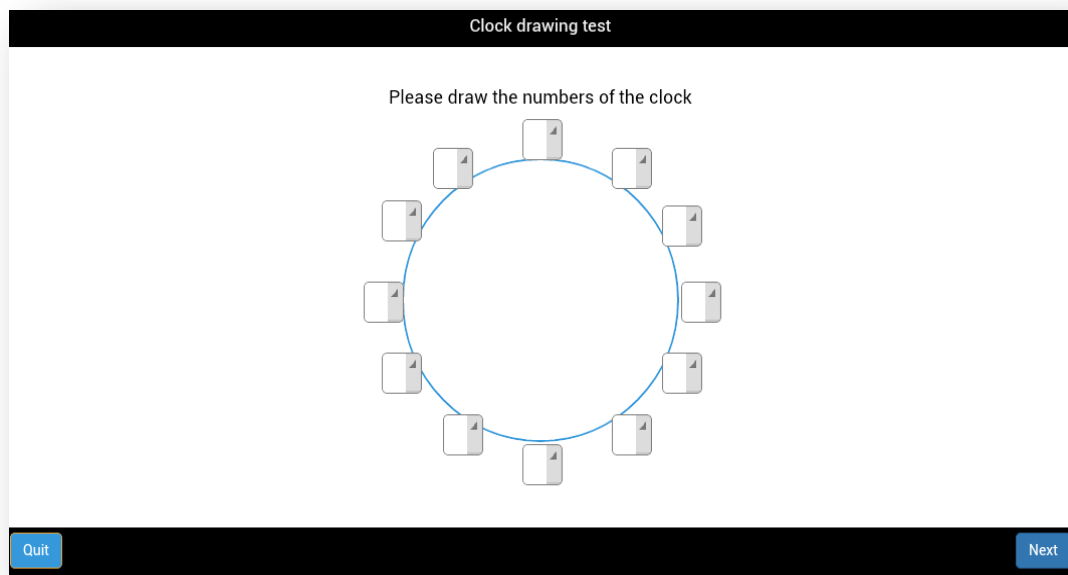


Figure 4.5 – Prompt to fill numbers in CDT

Filling correct all numbers should look like this:

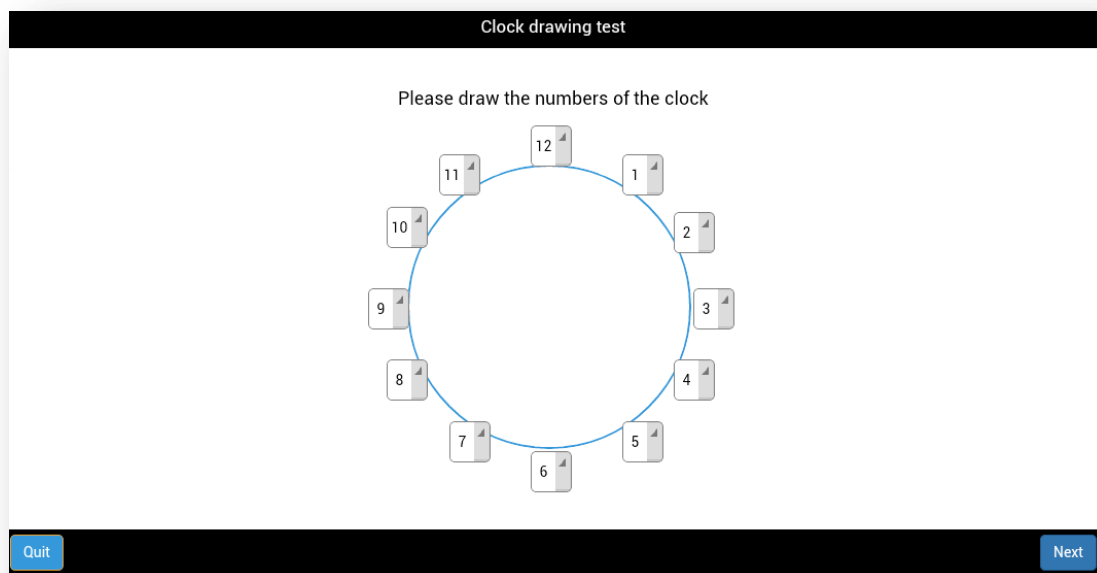


Figure 4.6 – Correct filled numbers in CDT

In the next phase two hands of the clock appear and the user should wriggle them in order the clock to show the correct time.

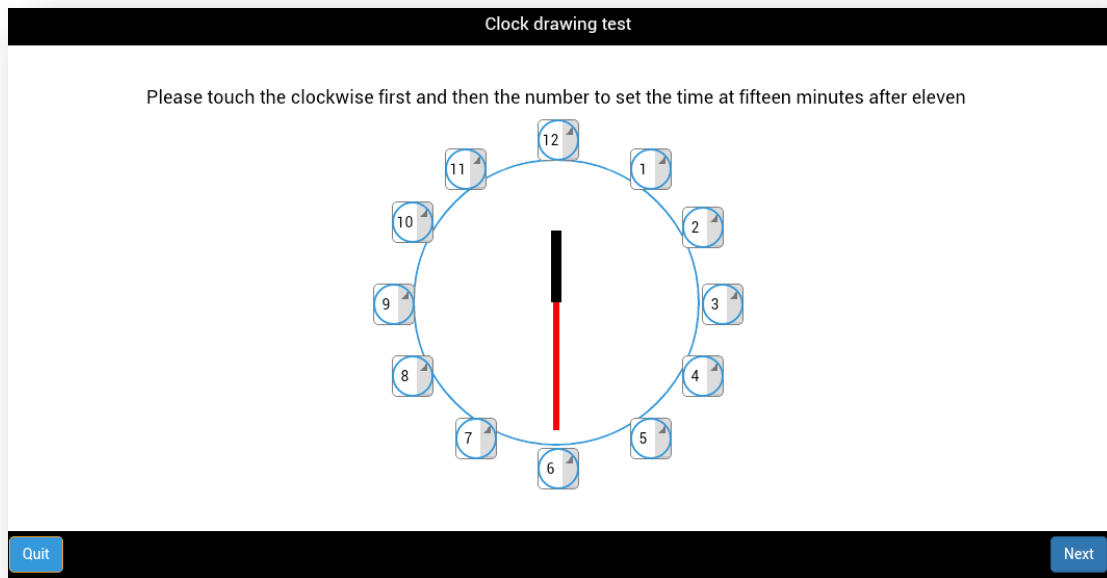


Figure 4.7 – Prompt to wriggle clockwise in CDT

Finally the correct clock drawing showing the requested time follows:

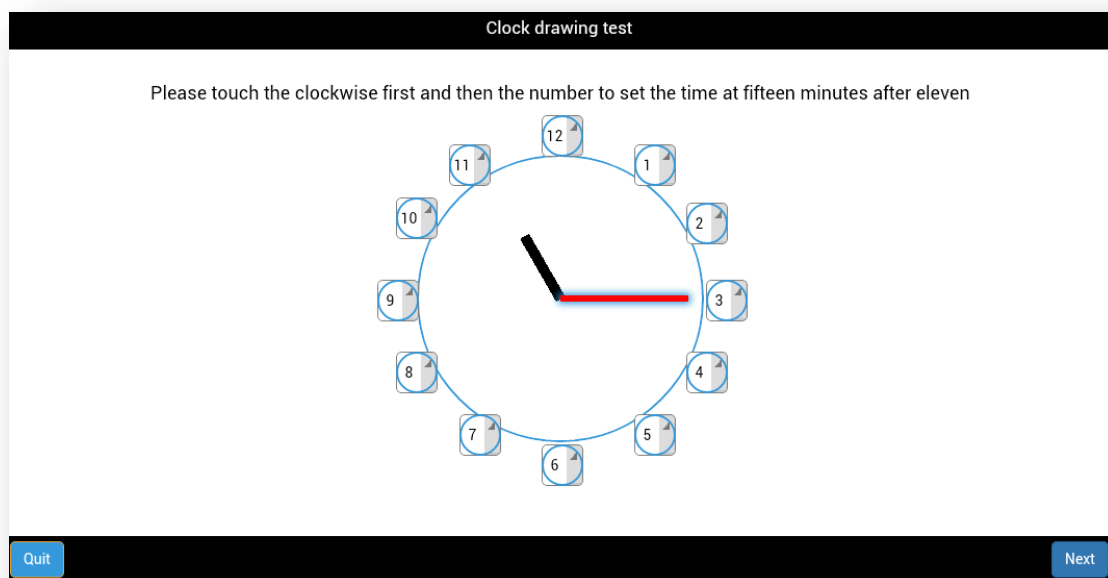


Figure 4.8 – Correct time in CDT

4.2.5 Mini-cog test

Goal

The goal is to recall 3 words given in start after the accomplish one the Clock drawing test (CDT).

Scoring

Total score is 7 points.

For each remembered word 1 point is given. The scoring of the CTD described in section 4.2.3 also counts here with its 4 points.

Screenshots

In this test we first ask the user to remember 3 words (not random - have been validated in a clinical study).



Figure 4.9 – Words to remember in Mini Cog

Then we prompt the user to do the Clock drawing test (CDT) as described in the previous section.

After the CDT is done we prompt the user to recall and select the words that was told to remember before.



Figure 4.10 – Showing words previously given in Mini Cog

The user simply tabs the words in order to select them and the test is finished.

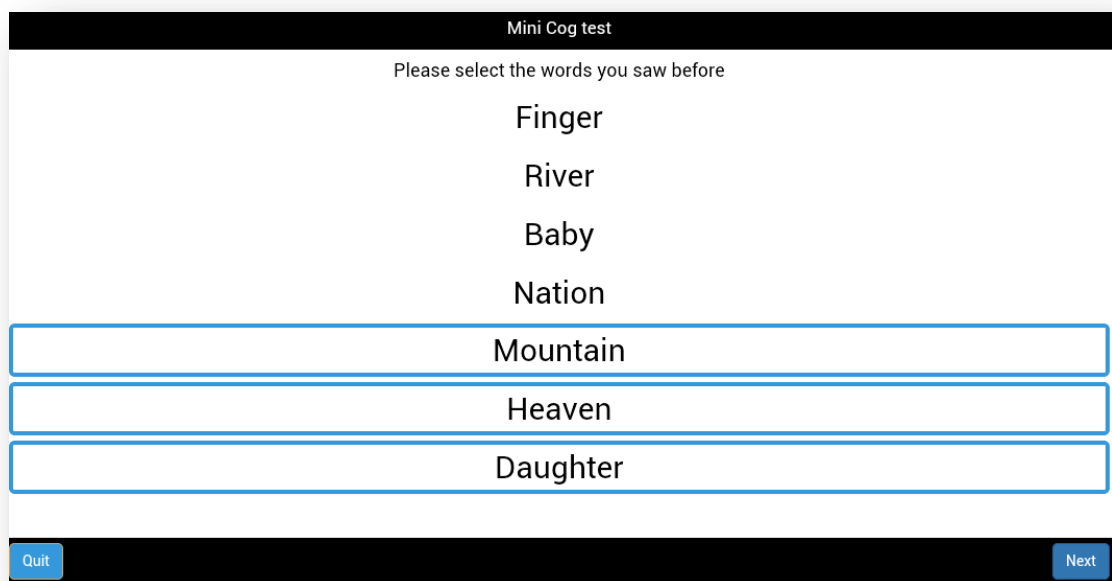


Figure 4.11 – Selecting words previously given in Mini Cog

4.2.6 MoCa Test

Scope

It assesses different cognitive domains: attention and concentration, executive functions, memory, language, visuoconstructional skills, conceptual thinking, calculations, and orientation.

Scoring – Screenshots

The Visuoconstructional Skills are shown through the Cube that contains itself the Alternating Trail Making.

The examiner gives the following instructions, pointing to the cube:
“Copy this drawing, (all edges and vertices must be included). Tap two vertices, one in a time to create an edge that connects them.”

Scoring: One point is allocated for a correctly executed drawing.

- Drawing must be three dimensional
- All edges must be drawn
- No line can be added

A point is not assigned if any of the above criteria are not met

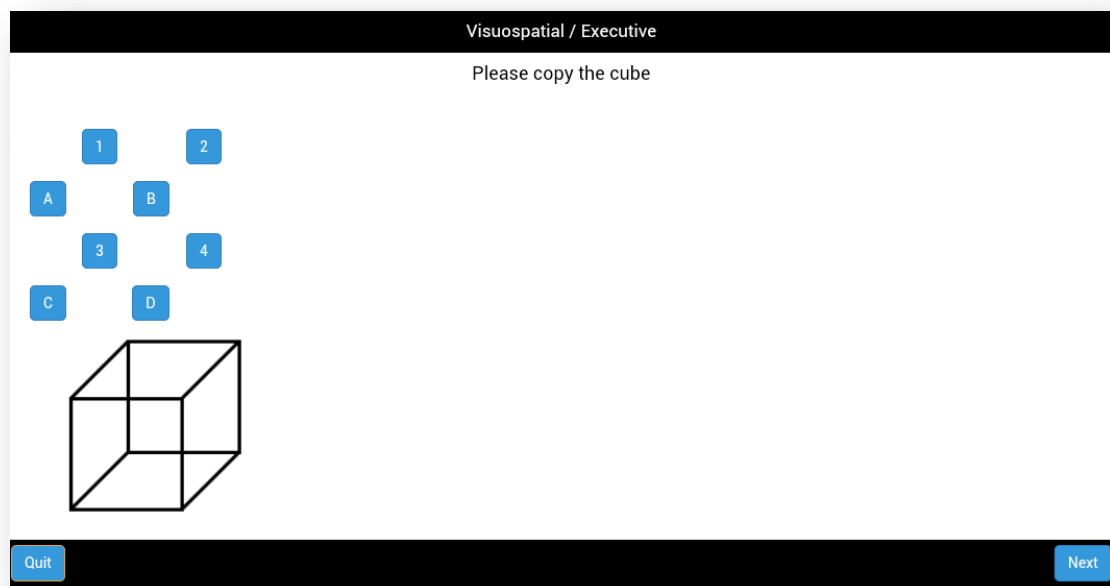


Figure 4.12 – Cube prompt to copy in Moca

As said in the instructions the user can draw an edge by taping the vertices that wants to be connected. When a vertex is taped then its color changes to orange in order to help the user see that is “active”.

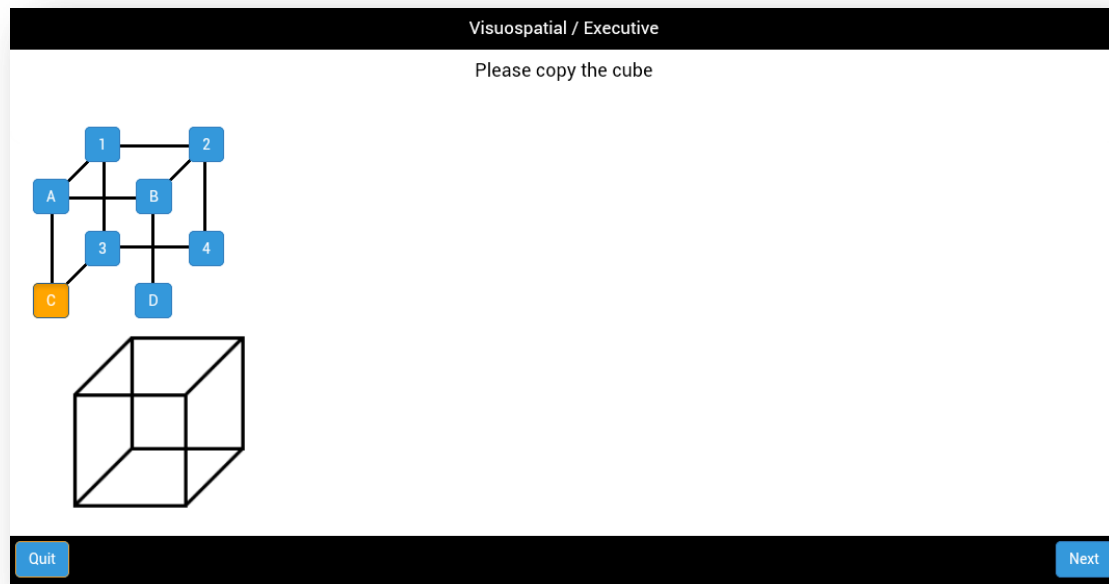


Figure 4.13 – Preceded drawing of cube in MoCa

When all edges are done the cube is successfully copied.

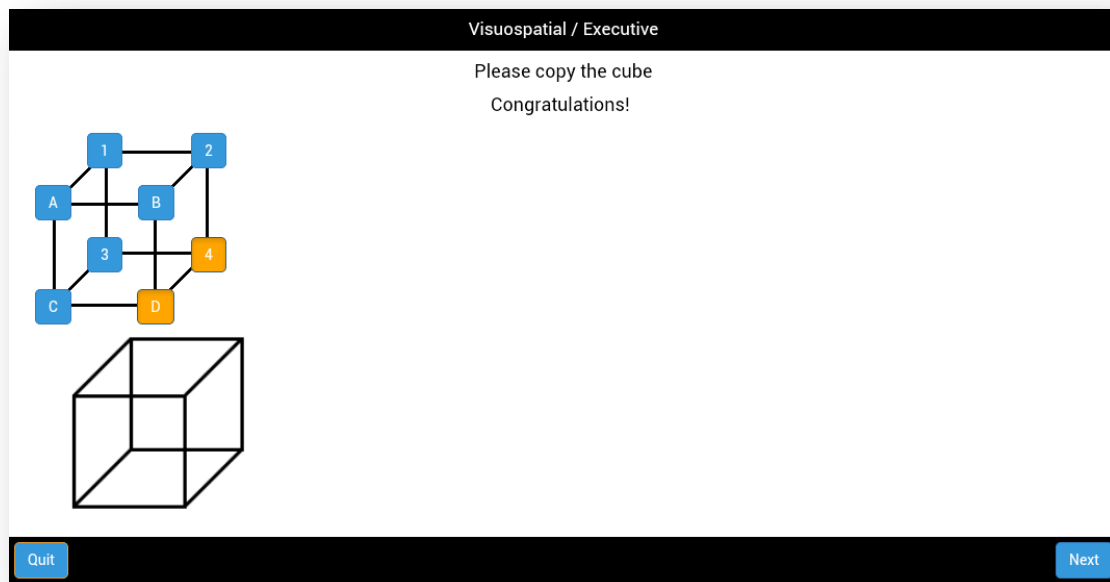


Figure 4.14 – Fully copied cube in MoCa

Short memory is checked through the following subtest of MoCa. A picture shows and in the next screen the user must tab this picture among many others.

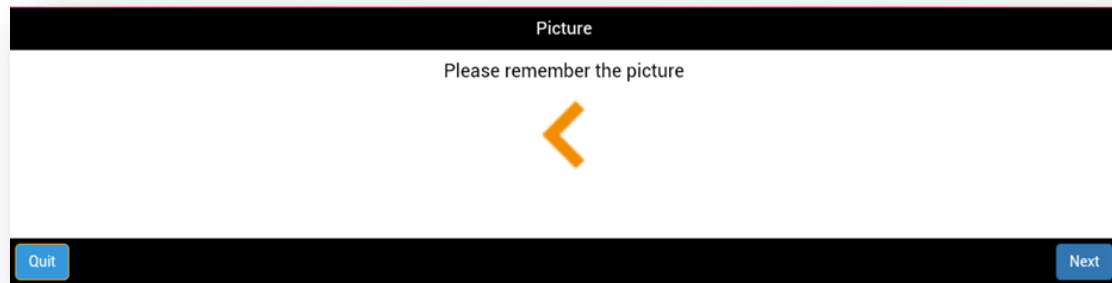


Figure 4.15 – Picture to remember in MoCa

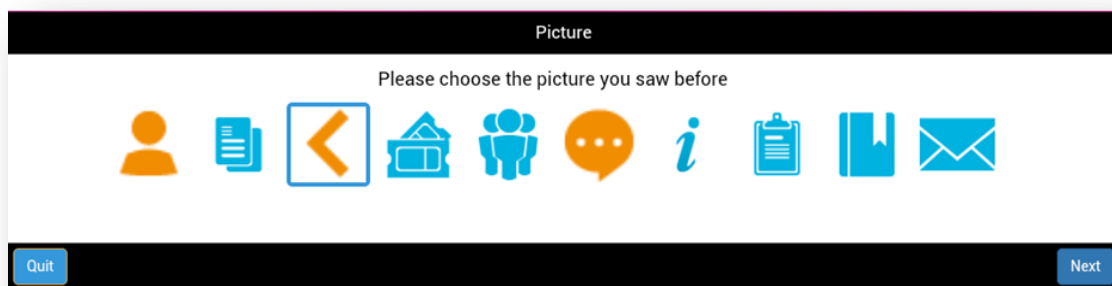


Figure 4.16 – Past picture to choose in MoCa

The CDT is the next test performed. After that the user should name the animals' name showing in photos.

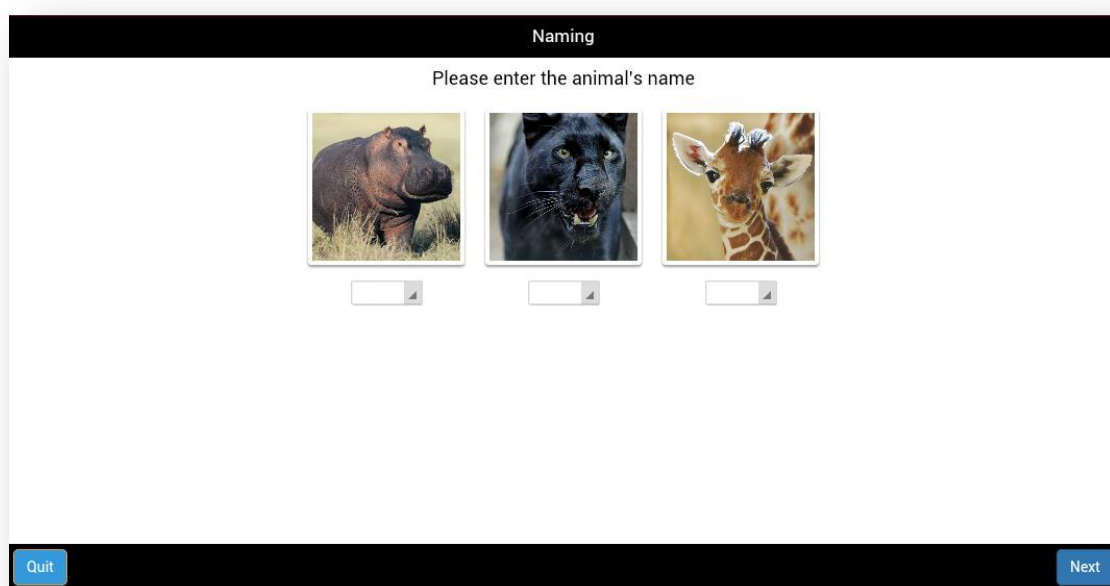


Figure 4.17 – Prompting to name animals in MoCa

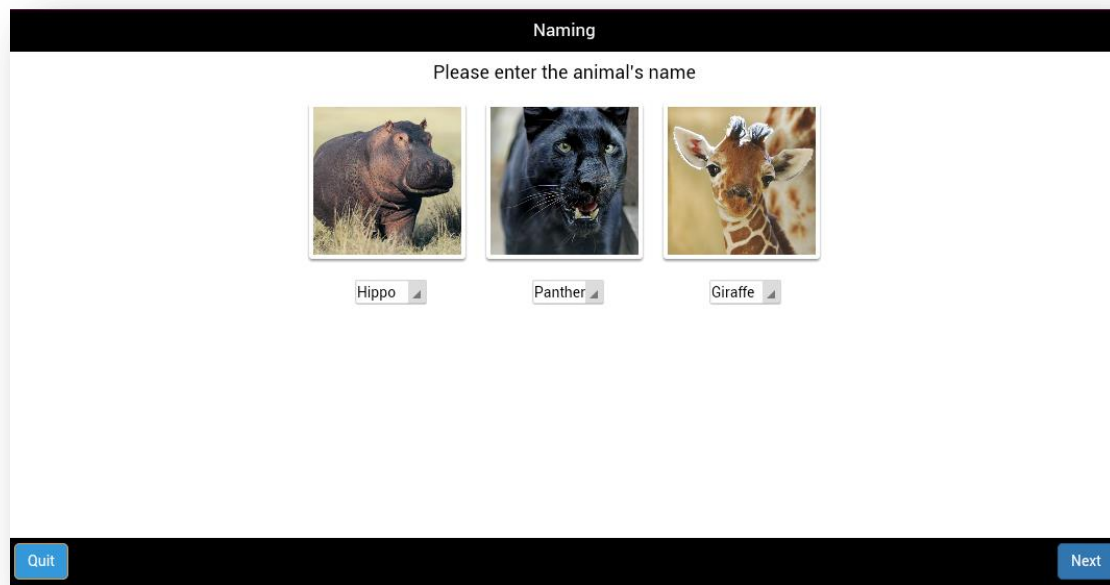


Figure 4.18 – Correct naming in MoCa

Long memory is checked by prompting user to memorize 5 words given. These words will be requested later.

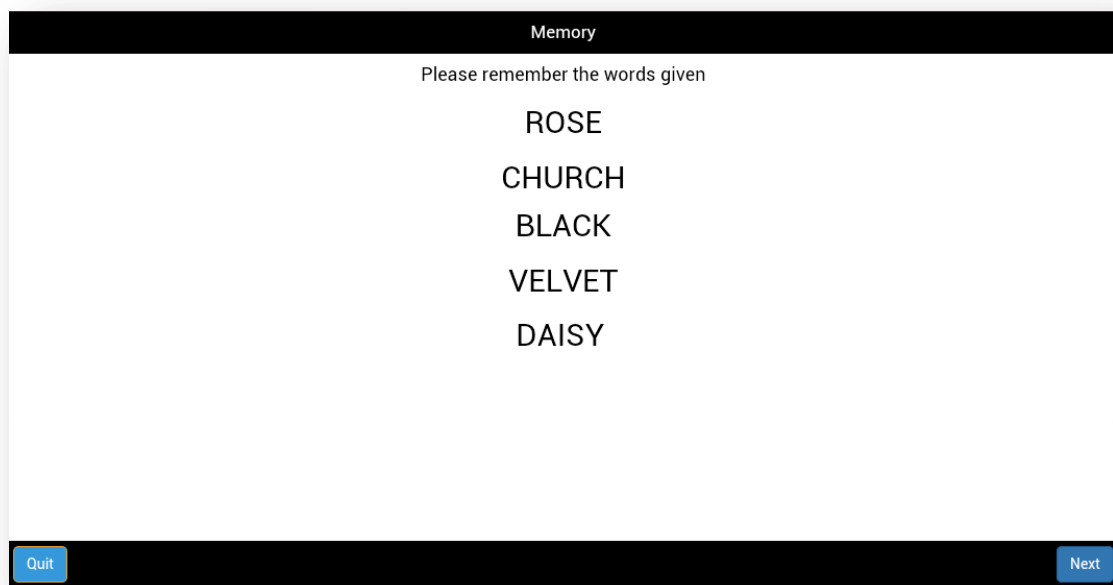


Figure 4.19 – Long memory part 1 in MoCa

User is prompted to sort numbers or letters in ascending order. Subtest scores only if succeeded the first time without any repositioning.

Attention

Please enter the numbers in ascending order

H B Q J W

Quit Next

Figure 4.20 – Attention subtest in MoCa

Next the user must combine triplets with categories. The categories are shown in a drop down box.

Abstraction

Please choose similarity for the words given

Juice - Water
Liquids

Banana - Apple
Fruits

Kitchen - Washing machine
Appliances

Quit Next

Figure 4.21 – Category choice in MoCa

After these tests the user now is prompt to recall the words given in part 1 Long Memory subtest. The procedure is the same as implemented in Mini Cog test. The user must tap the words in order to choose them.

Memory

Please select the words you were told to remember

RED

ROSE

YELLOW

BLACK

VELVET

CHURCH

DAISY

Quit Next

Figure 4.22 – Long memory part 2 in MoCa

Then the user must subtract 5 times the number 7 from 100. In order to gain points all the five subtractions should be defined.

Subtract

Please subtract 5 times the number 7 from the number 100

1. 93

2. 86

3. 0

4. 0

5. 0

Quit Next

Figure 4.23 – Subtraction part in MoCa

Chapter 5 – Conclusions

5.1 Usability – Hardware & Software issues

During usage obstacles with the devices were discovered. One of these obstacles included the touch sensitivity of tablets.

Based on the experience of participating patients, the greatest challenge was the accuracy of the touch interface.

Most mobile tablets use a capacitive touchscreen panel. Unlike traditional resistive panels that only require an object to exert pressure on the panel, usually a stylus; capacitive touchscreen panels require an electrical conductor from an object, e.g. fingertip, to respond. In addition, older adults, especially AD or dementia patients, lack the capacity to understand this difference. Many of the adult patients used the top of their fingernail when touching the device so they become easily frustrated with the application.

Application sounds and device feedback can either startle or confuse patients. Device dimensions and weight is also another challenge. During usage, most patients that participate were not able to hold the device independently. Many required a tablet or a stand to prop the tablet. The majority of patients complained that holding the device is rather heavy and interacting at the same time was not possible. In addition, screen size should not be below 10.2”.

All these obstacles concerning usage of mobile devices will be overcome in the future as young people now are very familiar with such type of technology.

The following changes were made in order to avoid problems occurred during the usage of the application concerning the software:

- We had to implement simplest graphics as most of the people did not concentrate and these rich graphics confused them. We also had to reduce the colors to 3 with a white background so the tests to be more clear and easy reading.
- We had to change the circle drawing event in the CDT. First of all we changed the threshold to 200 px due to the fact that the drawing of a cycle for the beginning to coincide with the end is very difficult using our finger. Also at the first implementation the line of the cycle drew in the screen was reflected but this made people rolling up their finger in order to check it, causing the loss of the full rotation event. To overcome this we do not appear the line unless we have a full round. At last we changed the “catch” time needed to make the circle as many users scroll their finger quite late.
- In the second stage of the CDT we realized difficulties in reading and putting numbers when a very small or a very big cycle was drew, so we independent that stage prompting the user to put the numbers on a fixed size circle.
- At a first place we had various types of inputs, numbers had to be drag n’ drop on the clock, words had to be written using the keyboard but all these had to change into drop-down boxes in order to avoid the misspellings that had nothing to do with the results of the tests, reduce the time needed to implement the whole test and to make the application more usable.
- Active vertex in the cube copy of MoCa test was not separated clearly from others causing problems to connect it with others. We changed active’s vertex color to orange and this problem disappeared.

5.2 Future work

In this thesis we developed a mHealth screening platform that combines various neurological tests which will help the diagnosis of early dementia and Alzheimer disease.

Using AngularJS models with JSON representation (see samples in appendix B) we give the opportunity to any future changes to be done quite

easy. Also the application is implemented in a way that any future tests (such as Modified Mini Mental Exam (3MS), General Practitioner Assessment of Cognition (GPCOG), Psychogeriatric Assessment Scale (PAS) and determination of tremor, early symptom of Parkinson's disease, through spiral drawing mathematical evaluation –Appendix C) could be added easy. We just add new partials for the view, controllers and models, built the updated project with Cordova and export it to the compatible desirable device.

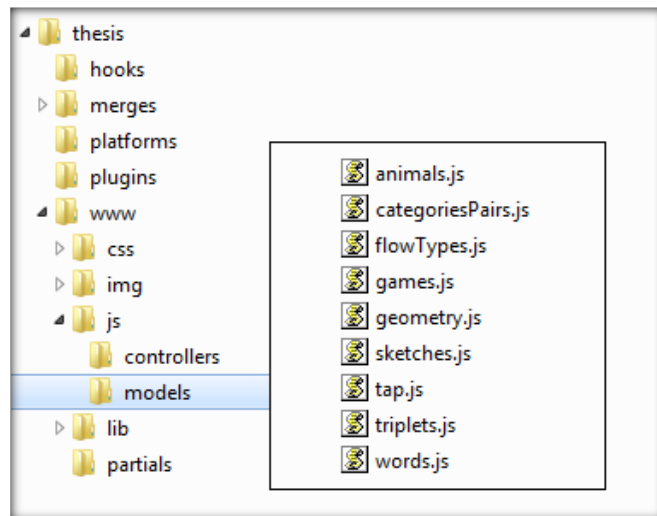
The application could use also an online database to store statistical data of tests. This way we will be able to find score without having the device the tests where first implemented. In addition CSV export, that is available with this implementation, could be imported locally in any mobile device with the application installed. Furthermore patients could perform tests anywhere and add the output data to the existing record.

Last the application might use stored sounds or record new ones in order to complete language skills tests or even for voice pattern recognition related to data need to be recalled.

APPENDIX A - Abbreviations

AD	Alzheimer Disease
ADT	Android Development Tools
API	Application Programming Interface
CDT	Clock Drawing Test
CST	Computer Self-Test
CNTB	Computerized Neuropsychological Test Battery
DI	Dependency Injection
GUI	Graphical User Interface
MBT	Mobile Cognitive Screening
MCI	Mild Cognitive Impairment
MMSE	Mini Mental State Examination
MOCA	Montreal Cognitive Assessment
MCT	Mini Cog Test
MVC	Model View Controller
OS	Operating System
PC	Personal Computer
SDK	Software Development toolkit
UI	User Interface
XML	Extensible Markup Language

APPENDIX B – MODELS SAMPLE SOURCE CODE



animals.js

```
var animals = [  
  {id: 1, name: 'Giraffe', photo: 'giraffe.jpg'},  
  {id: 2, name: 'Hippo', photo: 'hippo.jpg'},  
  {id: 3, name: 'Lion', photo: 'lion.jpg'},  
  {id: 4, name: 'Tiger', photo: 'tiger.jpg'},  
  {id: 5, name: 'Panther', photo: 'panther.jpg'}  
];
```

flowTypes.js

```
var flowType = {  
  CLOCK:1 ,  
  MINICOG:2,  
  MOCA:3,  
  HISTORY:4,  
  LOGOUT:5,  
}
```

games.js

```
var games = {  
  flow_clock: 'Clock',  
  flow_minicog: 'Minicog',  
  flow_moca: 'Moca',  
  animals: 'Animals',  
  categories: 'Categories',  
  clock: 'Clock',  
  triples: 'Triplet choose',  
  cube: 'Cube',  
  sketch: 'Picture selection',  
  words: 'Word memory',  
  hangman: 'Ordering',  
}
```

```
    subtraction:'Substraction'  
  }
```

words.js

```
var words = [  
  "FACE",  
  "VELVET",  
  "CHURCH",  
  "DAISY",  
  "RED",  
  "YELLOW",  
  "NOSE",  
  "HOUSE",  
  "FLOWER",  
  "SCHOOL",  
  "BLACK",  
  "SILK",  
  "ROSE",  
  "MOUTH"  
];
```


APPENDIX C – BIBLIOGRAPHY

- [1] *The Computer Self-Test (CST): A Computerized Internet Accessible Cognitive Screening Test for Dementia* John Dougherty, Jr., Rex Cannon, Andrew Dougherty, Lorin Hall, Jennifer Janowitz, Neuroscience Center; University of Tennessee Medical Center
- [2] *The Computerized Self-Test (CST): An Interactive, Internet Accessible Cognitive Screening Test For Dementia* 1928 Alcoa Hwy, Medical Bldg B, Suite 102. Knoxville, TN 37920. *Journal of Alzheimer's Disease* 20 (2010)
- [3] *Impact of Accelerometry and Spirography Data Analysis of Essential Tremor on BOLD-fMRI Data Interpretation - University of Miami Scholarly Repository-Saman Sargolzaei*
- [4] *Simple scoring of the Clock-Drawing Test for dementia screening* Ejnar Alex Kørner¹, Lise Lauritzen¹, Flemming Mørkeberg Nilsson, Annette Lolk & Peder Christensen, *Dan Med J* 59/1 January 2012 *DANISH MEDICAL JOURNAL*
- [5] Freitas S. et al. *Montreal Cognitive Assessment: validation study for mild cognitive impairment and Alzheimer disease. Alzheimer Dis Assoc Disord.* 2013 Jan;27(1) 37-43.
- [6] *COGNITIVE ASSESSMENT TOOLKIT- C.B. Cordell et al. / Alzheimer's & Dementia* 9 (2013) 141–150
- [7] Kansagara D, Freeman M. A systematic evidence review of the signs and symptoms of dementia and brief cognitive tests. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/21155200>. Accessed June 7, 2011.
- [8] Roalf et al. *Comparative accuracies of two common screening instruments for classification of Alzheimer's disease, mild cognitive impairment, and healthy aging. Alzheimer's & Dementia* Volume 9, Issue 5 , Pages 529-537, Sept. 2013.
- [9] Wang et al. *Montreal Cognitive Assessment and Mini-Mental State Examination performance in patients with mild-to-moderate dementia with Lewy bodies, Alzheimer's disease, and normal participants in Taiwan. Int Psychogeriatr.* 2013 Aug
- [10] Kane R, Kay G. *Computerized assessment in neuropsychology: a review of tests and test batteries. Neuropsychol Rev.* 1992;3:1–117
- [11] Salma S. Soleman Hernandez et al. *Apathy, cognitive function and motor function in Alzheimer's disease. Dement Neuropsychol* 2012 December;6(4):236-243.
- [12] Freitas S. et al. (2012). *Montreal Cognitive Assessment (MoCA): Validation study for Mild Cognitive Impairment and Alzheimer's disease.*

- Alzheimer Disease and Associated Disorders, doi: 10.1097/WAD.0b013e3182420bfe.
- [13] Markwick A, Zamboni G, de Jager CA. Profiles of cognitive subtest impairment in the Montreal Cognitive Assessment (MoCA) in a research cohort with normal Mini-Mental State Examination (MMSE) scores. *J Clin Exp Neuropsychol.* 2012 Aug;34(7):750-7. Epub 2012 Apr 3.
 - [14] *Designing Mobile Applications to support Mental Health Interventions* Mark Matthews, Gavin Doherty, David Coyle and John Sharry-Department of Computer Science, Trinity College Dublin.
 - [15] Munro, Jamie (March 29, 2012). *20 Recipes for Programming PhoneGap: Cross-Platform Mobile Development for Android and iPhone (1st ed.)*. O'Reilly Media. p. 76. ISBN 978-1-4493-1954-0.
 - [16] *Mobile Technologies heralding innovations in healthcare industry*-www.passbrains.com
 - [17] *Deloitte Center for Health Solutions, mHealth in a mWorld, How mobile technology is transforming Health Care*
 - [18] Marinacci, Joshua (March 21, 2012). *Building Mobile Applications with Java: Using the Google Web Toolkit and PhoneGap (1st ed.)*. O'Reilly Media. p. 86. ISBN 978-1-4493-0823-0.
 - [19] Lunny, Andrew (September 23, 2011). *PhoneGap Beginner's Guide (1st ed.)*. Packt Publishing. p. 328. ISBN 1-84951-536-0.
 - [20] *IBM Software Thought Leadership White Paper WebSphereNative, web or hybrid mobile-app development*
 - [21] Ghatol, Rohit (November 14, 2011). *Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5 (1st ed.)*. Apress. p. 700. ISBN 1-4302-3903-4.
 - [22] Myer, Thomas (December 13, 2011). *Beginning PhoneGap (1st ed.)*. Wrox. p. 336. ISBN 1-118-15665-X.F. Berkman, «How the Cellphone Got 'Smart',» 15 Oct 2012.
 - [23] Mudge, JT. *Native App vs. Mobile Web App: A Quick Comparison*. 2012
 - [24] *Applying Mobile Application Development to Help Dementia and Alzheimer Patients, Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 3rd, 2013*
 - [25] *International Conference on Computers Helping People with Special Needs. (2013, July 9). Computer-assisted augmentative and alternative communication (CA-AAC). Retrieved April 22, 2013*
 - [26] *Kinvey-native-vs-web-vs-hybrid* by Ishan Anand

- [27] *Mobile Applications for the Health Sector by Christine Zhenwei Qiang, Masatake Yamamichi*, Vicky Hausman, Robin Miller, and Daniel Altman-ICT Sector Unit World Bank-April 2012*