CHALMERS

Motionera Mera

A cross-platform application for physical exercise

Sebastian Eriksson
Aki Käkelä
Eric McNabb
Alexander Yeh

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2014
Bachelor of Science Thesis

Motionera Mera
A cross-platform application for physical exercise

SEBASTIAN ERIKSSON
AKI KÄKELÄ
ERIC MCNABB
ALEXANDER YEH

Examiner: Jan Skansholm

Chalmers University of Technology University of Gothenburg Department of Computer Science and Engineering SE-412 96 Göteborg Sweden Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Gothenburg, Sweden 2014

Abstract

The City of Gothenburg wanted an application for their current campaign promoting physical exercise for the citizens of Gothenburg. The work consists of digitizing their current card system where users make marks on the card when they exercise. This report discusses the various aspects of adapting the application for both mobile and desktop devices, as well as the server components required to manage the campaign and user data. The result is an application primarily built with web technologies, which makes it compatible with any device that has a web browser, including smartphones, tablets, desktops, and TVs. The discussion comprises the user interface and differences in the mobile and desktop platforms, primarily relating to Android and iOS on the mobile side. The discussion also involves security and privacy in the client-server communication.

Sammanfattning

Göteborgs Stad önskade en applikation för deras nuvarande kampanj för att främja fysisk träning för Göteborgs medborgare. Arbetet består av att digitalisera deras nuvarande kortsystem där användarna gör markeringar på kortet när de motionerar. Denna rapport behandlar olika aspekter av att anpassa programmet för både mobila och stationära enheter, samt att utveckla de serverkomponenter som krävs för administration och synkronisering av användardata. Resultatet är en applikation i första hand byggd med webbtekniker, vilket gör den kompatibel med alla enheter som har en webbläsare, inklusive smartphones, surfplattor, datorer och TV-apparater. Diskussionen omfattar användargränssnitt och skillnader i de mobila och stationära plattformar, främst avseende Android och iOS på den mobila sidan. Diskussionen handlar även om säkerhet och integritet i klient-server-kommunikationen.

Acknowledgements

We would like to thank our supervisor, Prof. Graham Kemp, for supporting us throughout the project. Johanna Tidholm and Martin Berglund working for the City of Gothenburg for making the project possible and for their assistance and feedback throughout the project. Åza Larsson, for the help we received with regards to graphics. Naomi Hoang, for having the patience to teach us how illustrator works. We would also like to thank PhD student Susanne M. Bernhardsson, Fredrik Hieronymus MD for their help with proofreading and suggestions for this report. Lastly, we would like to thank everyone who gave us feedback and advice during the project.

The Authors

Contents

1

Introduction

This chapter describes the background for the project and its purpose, and concludes with an overview of the report.

1.1 Background

In 2001, the City of Gothenburg began developing a promotion designed to encourage people to exercise more in their daily life. The promotion began simply as "Promenera Mera," or "walk more," but grew over time to eventually include other physical activities such as swimming and cycling. The promotion involves filling out cards consisting of 25 squares. For each day that the user exercises for 30 minutes or more they check one square. With the card filled, the user can mail the card to the City of Gothenburg to qualify for a chance to win a prize.

10,000 cards were submitted in 2013. The cards are received by mail, after which one is drawn manually; an avoidably laborious and time-consuming process. With the popularity of the promotion also in decline, the City of Gothenburg decided that an application, or *app*, would revitalize the campaign while making it more accessible to a broader audience.

Although the promotion is currently most popular with an older demographic, the application does not have the same target audience as the physical cards. However, the transition from physical cards to digital cards includes usability requirements for the product for those less versed in technology, but also brings interesting opportunities for an expansion of the promotion.

In line with the City of Gothenburg's strategy of reaching their goal with a mobile application there has been an extensive study[1] collecting data on exercise habits with the aid of mobile devices. The purpose of the study was to see if mobile devices and applications could be used for gathering data and what information could be extracted from that data. The study showed that several factors could improve exercise habits, some of which are mentioned in Table 2.1. It also showed that applications are a powerful way of gathering data for future studies.

1.2 Purpose

The primary goal of the project is to digitize the current physical card system by designing and building a functional mobile application prototype. By digitizing it, the application is set to reach a wider audience. Other goals include ensuring ease of use, added functionality, and more efficient administration. The application must also be available for the majority of users, meaning that it should be targeted at multiple platforms (Android, iOS, Windows, Mac).

The application focuses largely on creating a habit out of exercise by providing a logging system for physical activities. Currently, the most popular applications of similar nature, including Fitocracy[1], Runkeeper[2], and Nike+[3], tend to focus on utilizing GPS features, counting calories, or timing runs. Our application aims to instead serve as a casual and social exercise utility with a lower barrier of entry.

1.3 Overview

Chapter 2 specifies the goals and requirements of the application, and describes the planning that was made over the initial two weeks of the project. The chapter also explains the problems in the project, which are specified with key points described in some detail.

Chapter 3 describes the method by which the problems were solved. One of the main areas of the chapter is the choice of tools and frameworks, as they affect both the process

[1]`https://www.fitocracy.com/`
[2]`http://runkeeper.com/`
[3]`https://secure-nikeplus.nike.com/plus/`

and the end result considerably. The chapter also describes the subsystems and their communication, primarily the communication between client and server.

Chapter 4 presents a prototype of the final application. The chapter includes multiple figures showing the application from the perspective of the end user. It also describes some of the aspects that are not visible to the user, such as the client-server implementation.

Chapter 5 brings up some of the authors' own reflections and presents a critical evaluation of the project. The current limitations of the application are described and some suggestions of future work are presented. The chapter also presents the changes that could have been made with the experience gained from working on the project.

Chapter 6 concludes the report with some of the more prominent results of the project.

2 Problem specification

This chapter provides information about the features suggested by the City of Gothenburg and those suggested by the authors. The main categories are the functional and non-functional requirements, described briefly in this section. This is followed by a short section describing difficulties with cross-platform development. Concluding this chapter, the requirements concerning user interaction and the visual aspects are specified separately. This chapter contains all considerations made regarding the goals and design of the application.

The components that were eventually implemented in the resulting prototype are detailed in Chapter 4, and the process is described in Chapter 3. In other words, this chapter also describes features that were not included in the prototype, such as the leaderboard, the calendar, and the social features presented in Table 2.1.

Functional requirements involve tangible scenarios that the user should be able to do, for instance the ability to view news, announcements or log activities. Consequently, most of the client's requests are contained within this category. An example of a functional requirement is feedback upon selecting GUI[1] elements, which usually involves vibration on touch devices, or graphical effects on non-touch devices.

Non-functional requirements comprise the measurable and structural aspects of the application, such as scalability to different devices and graphical profiles, responsiveness to user interaction, and the security and privacy in handling personal data. It consists

[1]The GUI, or Graphical User Interface, allows users to interact with the system. It is composed of graphical elements such as buttons, input fields, and views.

of the aspects unseen by the end user, such as the maintainability and robustness of the code in the application.

Additional features originated both from early discussions with the client and later from suggestions by the authors. The client provided no designs for the application itself, but their suggestions are noted in Table 2.1 with an asterisk (*). In other words, the design of the application had not been established, meaning fewer constraints for the authors. The additional features were prioritized lower.

The platforms to be supported are both desktop and mobile devices. Since the platforms are inherently different some functionality must be implemented in multiple ways, such as notifications[2] or vibrations which are standard on the mobile side, but not on laptops or desktop computers.

2.1 Functional requirements

The main features requested were the capability to use the currently active promotion involving physical cards in the application, and the capability to view the locations of announced tracks and facilities. The application should therefore allow users to participate in the promotion by allowing the user to mark 25 exercise sessions, to view how many marks they have made toward a full card, and to submit a full card. Details about the lottery are provided in Section 2.2.8. Additionally, a map should be implemented with which the user can interact.

Additional features were added in order to provide more substance to the application. They were discussed with the client, but were prompted by the authors. The most demanding additional features planned are the leaderboard system and the news and announcements feature. Other goals discussed with the client are introduced and briefly described in Table 2.1.

[2]A standard way on e.g. the Android operating system to present concise information to the user. Our simulated notifications are presented in Section 4.1.11.

Feature	Description
Administrative interface	To handle events, facilities, and other administrative tasks. Also to show statistics about how the application is used.
Splash screen	The client wanted Gothenburg City's logo shown in the application. An unobtrusive location for it was the splash screen: the image shown while the application loads.
Map*	Shows the locations of Gothenburg City's exercise facilities, running tracks, and organized events. In addition, it should show the user's own position, their privately stored tracks, and tracks shared to them.
User profiles	Register users with shipment information, should the user win the lottery. Additionally, the user data storage must comply with the Swedish Personal Data Act (Personuppgiftslagen, PUL), as discussed in Section 2.2.7.
Leaderboard	Users can view personal statistics for exercises they have logged, and how they are doing compared to other users. Note: depends on the completion of user profiles.
Calendar	View past exercise sessions and plan future ones. To be combined with the training schedule.
Pedometer*	Counts steps taken by utilizing the sensors of the smartphone. Shown to improve the training experience [1].
Calorie counter*	Counts calories burned based on GPS calculations, achieved for instance by measuring distance traveled with the GPS feature or entered manually.
Social media	Interact with friends and share events through social media, e.g. Twitter or Facebook. Implementing such a feature has shown to increase frequency in training [1].
Training schedule	Schedule and view exercise sessions. Has been shown to improve adherence to training regime [1].

Table 2.1: Additional features planned for the application. Items marked with an asterisk (*) refer to features suggested by the client. These features are of lower priority than digitizing the card, and are ordered roughly in descending priority, taking an estimated implementation cost in mind. Although planned, not all of these features were implemented in the resulting prototype.

2.2 Non-functional requirements

The features presented in Section 2.1 lead to certain non-functional demands, including privacy and security issues. Some important aspects are the storage of personal data and the fairness of the lottery draw. Additionally, some usability aspects are important to consider, including responsiveness. Usability can be improved by building the application around the fact that mobile devices may lose connectivity while roaming.

The following sections introduce the categories of non-functional requirements that were considered at the start of the project.

2.2.1 Server

In addition to the application built for the end user, the project requires server-side components. This includes a database to store user data, scripts to handle interaction with the database, and an administrative interface to manage the database.

2.2.2 Privacy

Registering a user's information and personal data leads to security and privacy concerns. The application should therefore store the data in a secure manner. The user's personal details and activities should not be available to anyone but friends they have added and the system administrator. The system administrator should only have access to the information about the user that is required to moderate and maintain the application.

The usage statistics mentioned in Table 2.1 should reflect the total population of users and not be presented in a way that discloses the activities of any individual user.

2.2.3 Security

The application will store and transfer personal details about the users and their activities. It is therefore important to ensure that the data transferred between the application and the server, where it is stored, is secure. As described in Section 2.2.2, user data should only be available to the server administrator if it is required to moderate and maintain the application.

Users will register information consisting of their name and address information but also their email and password details. It is important that passwords are kept secure even if they are breached. This is important to ensure that the user's personal information and activities are not compromised, seeing as users are commonly reusing the same personal details with the same password.

2.2.4 Portability

An important point to consider is what data are accessible without a server connection. A practical example is storing the latest copy of some elements retrieved from the server when a connection was last established, such as news and announcements, but also user-specific data. The portability problem introduces subtle challenges in synchronizing user data. The user should be allowed to save user data while the device is offline. Similarly, the user should also be able to access their stored data in an offline state.

In short, the application should be as usable as possible even without a server connection.

2.2.5 Synchronization of user data

The application should allow the user to access their profile content from both desktop and mobile devices. It should also allow for concurrent sessions from these devices, as the user may be using the desktop version while making changes to the user data from their phone. A practical example is that the user should be able to plan their routes at home on a desktop device, then use the data on a mobile device.

The synchronization requires the data to be stored on a remote server to ensure that there is a single, reliable set of data that the clients retrieve. This means that there is only one place where the data stored are verified to be correct.

As an example, when the user views the activity logs, the application should attempt to update the list. If the device is not connected to the server, the update will fail. The locally stored copy of activities should then be displayed. If the device is connected to the server and the update succeeds, the device should have new data to overwrite the locally stored copy. The implementation of this is explained in Section 3.6.1.

2.2.6 Language of the user interface

The application will potentially be used by people with diverse mother tongues who may not be proficient in Swedish or English. Therefore, the application should rely on visuals over text as much as possible. In cases where text must be used, the City of

Gothenburg required the application to solely support Swedish since they did not want to discriminate against speakers of other languages by not including their language as well.

2.2.7 Swedish privacy law

Because the application will store personal details, the data have to be stored and managed in accordance with the Swedish Personal Data Act, as mentioned in Table 2.1. Since the application will ultimately be delivered to and used by a government agency, there are several important considerations, including the governmental handling of personal data [2].

Upon registering a user profile, the user must first agree to the terms and conditions of the application which state how their personal details are to be stored, managed, and what they are used for. Additionally, the application should also inform who is responsible for the storage of personal details together with contact information to this party. Users should be informed of the purpose of storing the information, and of the possibility for any individual user to retrieve the information stored.

The personal details should not be accessible by anyone, not even people within the City of Gothenburg. Only the people who have been assigned this responsibility should have access. The personal details have to be protected, and there are strict requirements on the authentication if the data are classified as sensitive, according to 13 § PUL [3]. As the data stored by the system does not have that classification, there are no specific requirements on the authentication.

2.2.8 Lottery

One of the main points of the application is that all registered users will participate in a lottery as soon as a card is submitted. Since the promotion is to be made into an application, this process will be automated. When the user reaches 25 logged activities on unique days and connects to the server to authenticate, they should automatically be made eligible to win a prize. Each set of 25 logged activities, i.e full card, results in an additional chance to win a prize in the quarterly lottery. A user can participate with a maximum of four submissions during each lottery period.

For the lottery to be fair it is required that the algorithm for choosing a winner will have to adhere to Swedish law, more precisely the lottery law (sw. Lotterilagen). The algorithm should therefore pool the submissions and randomly select a winner. The data in a submission should also be anonymized, which can be accomplished by encryption. The lottery law handles who is eligible to hold a lottery which is outside of our control.

2.3 Platforms

The application was requested to work for both desktop and mobile platforms, in particular Windows and Mac OS X on the desktop side and Android and iOS on the mobile side. Desktop support suggests supporting a large portion of the desktop users, and therefore a variety of desktop browsers, including Google Chrome, Firefox, Internet Explorer, Safari, and Opera.

The administrative interface requires only desktop support.

2.4 User experience

Beyond the functional requirements, the user interaction aspect was of particular interest, as the client had specified that they had a wide user base. The application should not only be appealing to a younger demographic but also an older one.

2.4.1 Graphical user interface

The application needs to run smoothly, without inconsistencies, on multiple devices and versions. Therefore, the user interface (UI) must be responsive: icons and text must be similar in relative size across various devices and therefore scalable [4]. The client presented a graphical profile, which they wished to have included in the application. The implementation process was not specified, but it was important for the City of Gothenburg that their graphical profile was adhered to. A challenge was to develop a design that would represent the same look and feel in the application as in the rest of the provided material. The clients did not have a clear picture of what they wanted from the application and needed assistance with the design.

2.4.2 User interaction

Beyond the visuals of the interface there are several design aspects which are important, especially when considering users of different age. Special regard has to be made with consideration to the older demographic, such as ease of understanding in how to use the application. Use of text is also to be minimized in order to give more space for other elements. Icons were the primary elements used in the space given, particularly due to the small screens of mobile devices. They were also designed to be as universally understandable as possible in order to remove much of the language barrier, as described in Section 2.2.6.

Another enhancement is to utilize the senses of hearing, movement, and touch, which can be achieved by making use of sounds, animations, and vibrations, respectively. When the user selects a graphical element on the interface, the use of vibrations on mobile devices act as an indicator of the application registering the user input. The application should therefore utilize senses other than vision, when appropriate.

2.4.3 Responsiveness

One of the early goals for the project was to make the interface responsive. This means minimal use of animations, especially those that cause delay, such as page transitions. This is not to be confused with the term *responsive design* which refers to adapting the visual content to the available space in the browser window.

An early concern regarding responsiveness was the use of web languages instead of native code[3] as web code introduces lag and other issues, as explained in Section 3.3.

[3]While C and C++ are native to Android, Google does not recommend using those languages for Android development [5]. Java will be referred to as native as it is the recommended language to use.

3

Implementation

This chapter outlines the process of solving some of the goals of Chapter 2 and provides details regarding the implementation. The first section describes the initial plan which was divided into three milestones. The plan is followed by a description of the choice of tools and frameworks and the motivations for using them. This is followed by an overview of the system architecture, summarized in Figure 3.1. The chapter concludes with some considerations about the topics introduced in Chapter 2.

3.1 Work process

The process involved two meetings with the client, the first of which was about the goals for the project, and the second about the status halfway. Additional meetings were difficult to plan due to scheduling conflicts. This led to more initiative being taken by the authors.

Each meeting was initiated with three short questions to each team member which contributed to a better understanding of the project as a whole:

1. What have you done since our previous meeting?
2. What are you planning to do until the next meeting?
3. Any impediments or stumbling blocks?

These questions are a component of the Scrum[1] software development method.

The development followed the cyclic development of Scrum, i.e smaller parts of the whole were completed iteratively. The idea of creating cycles in development is mainly to facilitate changes occurring during development, but as meetings with the City of Gothenburg were few and far between, significant changes rarely took place and the development method was changed. Eventually, the work process consisted simply of a list of tasks with assigned responsibilities, although a group meeting agenda was regularly kept. While Scrum involves additional development strategies such as team roles, they were not followed in this project.

3.2 Milestones

As Scrum was used early on, milestones were made to correspond with the end of each Scrum sprint. A sprint duration of three weeks was chosen, as it led to three milestones, each with clear main objectives from the backlog of features for the product. These milestones are summarized with their most important features below, in chronological order.

1. Digitize the card
   To achieve this goal there are several requirements to fulfill. Firstly, the groundwork for the system architecture had to be developed. Secondly, a UI was required to enable the user to access the promotion and interact with the application. Finally, a database was required to handle the storage of user data.

2. Statistics tool and calendar
   For the statistics tool to function there must be support for it on the server side. Past activities must be saved. There should be support for individual statistics, for instance the number of times a user has been running over the last year. There should also be support for comparing users with each other for a competitive aspect.

   The calendar is strictly graphical and requires similar functionality from the database as the statistics tool. The calendar should give a simple overview of past exercise sessions for the previous week, month, and year. It should also be possible to plan future activities and add reminders. This has been shown to be positive for the user's physical health as it increases motivation to exercise [1].

3. Map and schedule
   A possibility to view where the City of Gothenburg's exercise facilities and running tracks are located was requested. A map that can show where the user is and where

[1]https://www.scrum.org/

the facilities are located requires geographical coordinates of the facilities. There must also be server-side support for adding new points on the map through an administrative interface. The coordinates must be retrievable from the server.

For added usefulness of the calendar a simple training schedule should be combined with it. For example, a user who wants to run the Gothenburg Half Marathon may want to know how to practice for it. The schedule function requires the calendar or a similar interface to display the planned exercise sessions. It also requires space for storage in the database for the different schedules. This will stop the application from taking too much memory as the number of schedules increases. It will also enable multi-device access to user data.

3.3 Cross-platform development tool

In the initial phase of the project, different development options were researched. The main attribute sought was performance, since the requirements in Section 3.7 stated cross-platform support. The IDE[2] Titanium Studio was initially chosen, largely because of its platform-oriented design. The IDE would allow for development closer to individual platforms, such as coding in Java for Android instead of coding in JavaScript for web browsers. Since JavaScript is an interpreted language, meaning that it is not compiled before run, it has lower performance than, for instance, Java.

Using Titanium Studio, the final product would benefit from being faster and having better support since standard features and code, specific to the platform would be used. In other words, the code would be optimized for the particular device, as opposed to the browser running on the device. The IDE also packages several useful enhancements to the web programming languages including the MVC[3] framework *Backbone.js*, which in Titanium Studio is relabeled Alloy.

From previous experience, one of the authors suggested the mobile development framework PhoneGap[4]. It was briefly tested in the first week of the project, in fact before Titanium Studio. By the time the planning was finished we nonetheless transitioned to using Titanium Studio, as the advantages it promised made it considerably more appealing. Most of all, the native code would mean greater responsiveness and reliability. The stability and the low number of bugs from native coding were highly valued, making the decision to use Titanium Studio easy.

[2]Integrated Development Environment, generally a text editor which incorporates a compiler, but often much more. A popular example is Eclipse.

[3]The model-view-controller design pattern separates code by concern, such that graphical, functional, and data sections are separate.

[4]A mobile development framework. It started out as a means for web developers to create iOS applications without knowing Objective-C.

It was quickly discovered that Titanium Studio suffered from several problems, including, long compilation times and various errors with the emulator. The tool never worked well enough to justify its use: the authors all had different, well-known issues hindering usage. After re-evaluating the options, PhoneGap was reinstated as the primary development tool, as the research into using the tool yielded mostly positive results. The primary sign of comfort in using PhoneGap was its relatively long history, dating back to 2009, which is a long development period within the field of mobile phones. A drawback was its reliance on web technologies as opposed to the more native orientation of Titanium Studio. This lead to the applications built with PhoneGap considerably slower than native[6].

As PhoneGap was chosen as the development framework there was no IDE dependency, so the authors were free to use any development environment. The IDEs have no significant effect on the final product other than in the process of developing the application. Development was nevertheless made easier by the use of integrated tools such as debuggers and built-in code convention checkers. JavaScript allows for optional semicolons in code, and there is no single standard in, for instance, the naming scheme of variables. Hence, a code convention checker would greatly help with standardizing the code.

3.3.1 iOS development tools

Development for iOS is only available for Macs, i.e. an Apple computer, running its operating system OS X with Xcode installed. Xcode is an IDE required to develop for iOS devices and other OS X-specific applications, but can also be used for other types of software development. To be able to run applications on an iOS device, an Apple developer's account is required. Xcode proved to be quite useful, as it would detect specific errors in the iOS build, and also give informative feedback about the benchmarking of iOS devices such as memory utilization. Only one of the authors had previous experience with Apple devices. Since iOS development requires Apple hardware, this resulted in limited bug testing and development on iOS devices.

3.3.2 Emulation

Mobile devices have a wide variety of resolutions and screen sizes, and therefore different pixel densities. The smartphone simulation plugin Ripple[5] was used to test the application on different devices, mainly to ensure that the GUI was consistent throughout various devices.

The process of scaling the graphical components is discussed in Section 3.8.4.

[5]`http://emulate.phonegap.com/`

3.4 Server tools and framework

The requirement on the server-side tools and framework were that the application should be able to run on a web server. This is in order for the client to communicate with the server but also for hosting the administrative interface and the client interface for resetting the password. Additionally, it should have a model-view-controller (MVC) structure. It would also be of importance that it was scalable, well-tested and documented.

Since the client already had a web host set up on Loopia[6] supporting PHP and ASP.NET our choice was between those technologies. PHP is the most widely supported server-side scripting language since it is running well on Linux servers and there are more readily available frameworks and libraries. Thus PHP was chosen, and naturally MySQL[7] with it, since it is well integrated with PHP and supported by most web hosting solutions. It then followed that the server would have to support PHP version 5.4 or later, due to OpenSSL dependency as discussed in Section 3.10.2.

It was deemed useful to have one system divided into multiple controllers for handling everything instead of having separate systems. This was mainly to avoid code duplication as the administrative interface and server then could share the same codebase and primary models. This also has the advantage of providing consistency in not having multiple systems and codebases performing different tasks. Dividing into multiple systems could have performance advantages. Based on the number of requests the possible gain in performance would however not be sufficiently significant.

Partly due to a limited time span, but also for security reasons, as described in Section 3.10, an open source framework was sought rather than developing a custom framework. Based on the above requirements, CodeIgniter[8] was determined to be the most appropriate. It fulfilled all of our requirements and is widely used and thus tested and secure.

3.5 External code

Some of the initially desired features were discussed without regard to the technical difficulty in implementing them. Some of the features were also large enough to merit their own project. A solution to this problem was to use external modules and frameworks. External modules provide purpose-built APIs, which are programming interfaces that specify how applications should communicate with them. The APIs allow for more rapid

[6] `https://www.loopia.se/` A web host company
[7] MySQL is an open source database management system (DBMS)
[8] `http://ellislab.com/codeigniter`

progress, especially in the case of cross-platform development, as they provide complete and tested solutions to common problems.

External code was also used to expand the programming possibilities. In this section, we present the frameworks, libraries, and plugins that the application builds upon. Frameworks can be viewed as large-scale modules for the architecture of an application. Libraries are collections of utilities or smaller modules. Plugins are small-scale utilities, usually with a single, clear purpose. A plugin may, for instance, reformat dates or provide a simple element such as a time picker[9]. Although the developers of jQuery call it a library [7], it has a significant impact on the resulting application, as described in Section 3.5.2.

3.5.1 MIT License

All of the chosen external modules are under the MIT license, which states that the author of the work gives permission for anyone to use, change, distribute or sell the author's work in any way they choose [8].

3.5.2 jQuery Mobile

The jQuery Mobile library was used as it enables simpler scripting and expands the possibilities of manipulating HTML elements. The library is so extensive that the application is built around it: each page in the application is defined as a jQuery Mobile page, and navigation between them is also handled by the library. Some of the inherent features of the library are events, e.g. mobileinit, which tells the application that the mobile code has been loaded, and various page events such as pagecreate or pageshow. In order to execute code only when the page is first created, for instance, one would bind code to the pagecreate event. During development, these jQuery Mobile-specific functions became integral to the application. As a result, porting the application to other frameworks would be more difficult, unless equivalent features were to be provided.

jQuery Mobile also offers extensive, mobile-adapted style sheets. These change the standard layouts of regular elements, such as lists and buttons, to make them both more appealing and better suited for mobile devices. The issue with styling frameworks is that they regularly cause unexpected stylings as they apply to practically every element defined. Some problems include arbitrary thin lines across pages, tab bars either being shortened or stretching beyond the viewport, and elements such as radio buttons flashing upon selection. These problems are usually solved by excluding certain styles in the

[9]An element for selecting a duration or a time, e.g. in an alarm clock.

jQuery Mobile style sheet files. Nevertheless, there is a good reason not to alter the files: as the library is popular, it may have already been downloaded on the user's device, in which case it will be available for use immediately.

As jQuery Mobile provides a large set of style sheets, they often override those written by the programmer. In order to prevent this, various alterations have to be made to the specificity[10] of the styling rules, e.g. using identifiers (IDs) to trump the jQuery Mobile-defined rules. A good practice is to avoid using the *!important* tag in CSS, but the work in overriding the styles of the framework may sometimes be greater than the work in following the standards. If the framework is modified and supplied with the application directly, the problem of specificity disappears as the programmer simply removes the rules causing the unwanted overrides. In this project, specificity was a considerable issue to work with, as the choice to leave the framework unmodified was made to reach a greater potential with the application. One advantage is that, by using a URL to access the style sheets, the application can automatically apply the latest updates without the developer updating the application.

The use of jQuery Mobile during the development process is further discussed in Section 3.8.3.

3.5.3 Mobiscroll

The scroll wheel plugin implementation was a fairly simple process. The biggest problem was finding it as most scroll functions used in HTML and JavaScript are just up and down arrows. The advantage of the scroll wheel is its simplicity of usage, which is in line with the goal of being easy to use for a wide audience. The scroll wheel plugin found was MobiScroll[11].

Mobiscroll is an entire library based on JavaScript and CSS that supports multiple platforms. The Mobiscroll library provides several choices for selecting a duration, which was a function sought for logging exercises. Only a light white android theme and time function were used with slight modifications to fit the application better.

The scroll wheel allows the user to swiftly choose a duration for the exercise by spinning the wheel by touch. The wheel responds intuitively with inertia, slowly decelerating after the user has finished the swiping[12] motion.

[10]The list of rules can be found in `http://cssspecificity.com/`.

[11]`http://mobiscroll.com/faq`

[12]Swiping means moving the finger quickly across the touch surface and releasing.

3.5.4 Map

It was difficult to find a plugin which offered the functionality that was sought. The map function is a very useful feature, no matter where one is by simply opening an application a person can find out where they are and how to get to where they are going. Because of the great benefits of maps in smartphones there are a large number of solutions available.

When looking for which plugin to use there were several requirements. Firstly markers were needed to indicate where the City of Gothenburg has its facilities. Secondly, the map feature needed to work for all platforms. Lastly, it needed to be reasonably easy to implement considering the timeframe for this project.

The different options were jQuery's Google Maps plugin, Google Maps API by itself and Bing Maps API. Both Google Maps and Bing Maps were estimated to take up more development time than the chosen jQuery Google Maps plugin. The problem with Google Maps API was also that Google has a restriction on the number of users, which could become a problem in the future. One way to solve this is to store the map coordinates in the database and load it from there. This removes some functionality, as it becomes less dynamic and geolocation and zooming might be affected negatively.

3.5.5 Other plugins

The application uses plugins to enhance both the user interaction and the visuals. The news and announcements page utilizes two plugins for the time stamps on the news posts: *moment.js*[13] converts the times to display as differences in time, such as *five minutes ago*, while *livestamp.js*[14] extends this feature by providing live updates of the stamps.

The WebKit[15] engine running on mobile devices will, by default, introduce a delay of 300 milliseconds for the simulated touch events [9]. This is because of the existence of double tap events: the system must wait to be able to determine whether a single tap or double tap event was given [10]. Double tap events were not required in the application, so the touch delay proved only to be a burden. The *fastclick*[16] plugin was used to effectively remove the delay.

[13] `http://momentjs.com/`

[14] `http://mattbradley.github.io/livestampjs/`

[15] A web page renderer used in the Safari web browser, previously also used in Chrome.

[16] `https://github.com/ftlabs/fastclick`

3.5.6 Server-side modules

The CodeIgniter framework includes a large set of available libraries. The application is using the database and session libraries. The database library manages the connection to the database and allows the application to construct and run queries, but also provides methods to create custom transactions for synchronization of queries. The session library provides a set of methods to create and manage sessions which are used in the administrative interface in order to verify that the administrator is logged in. Additionally, an authorization library was developed to handle the token[17] authorization as discussed in Section 3.10.

3.6 System architecture

The system architecture is composed of several smaller systems, largely divisible into two categories: the client side and the server side. These, in turn, also make up smaller systems. For an overview, see Figure 3.1. The code of the application is structured by separation of concern, that is, smaller sections manage only a limited part of the whole problem. This is to make the system more manageable.

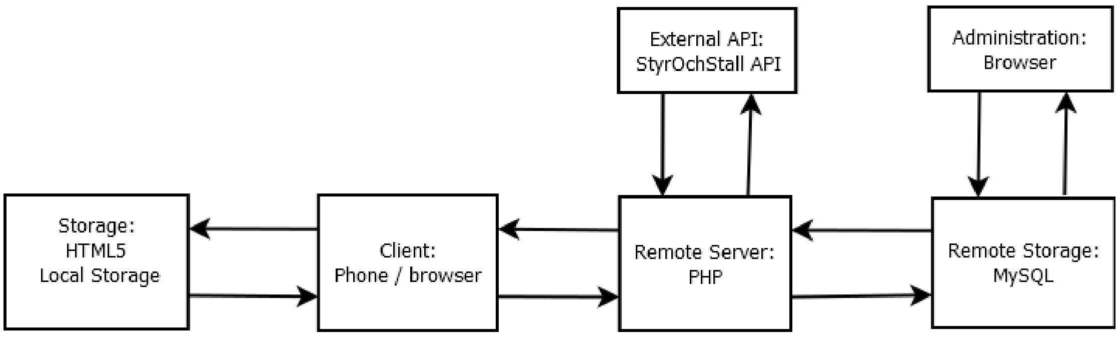


Figure 3.1: An illustration of the communication between subsystems. For example, the client makes use of the HTML5 Local Storage API, as detailed in Section 3.6.1.

As seen in Figure 3.1, the client stores data into its associated HTML5 local storage, which is then synchronized with the remote server to ensure that the data on the remote storage is correct.

The remote server communicates with external APIs which consists mainly of APIs from the City of Gothenburg's so called open data[18]. The data the server retrieves are stored temporarily on the file system. The data are then parsed and stored in the remote

[17]A token is a string or key that is used to receive authorization to perform certain operations.
[18]`http://data.goteborg.se/` Gothenburg open data

storage, i.e. the database. This data can then be retrieved by the client directly from the server. In this way, the server works as a cache for the external APIs.

The remote server also has an administrative interface that is sharing the same codebase as the server but is separated in that it is only accessible by the system administrator. The administrative interface is what allows the system administrator to maintain and moderate the system and data, as discussed in Section 4.3.

3.6.1 Persistent data with HTML5 Web Storage

The application uses the HTML5 Web Storage API to store persistent data for the user. The API consists of storing and retrieving data in a key-value pair. User data includes activity logs, log in details in the form of a token, and the news and announcements.

The idea behind storing data on-device is to provide offline access to the latest information. As such, the server is responsible for maintaining a correct copy. This has the added benefit of enabling users to use several devices, most importantly both the desktop and the mobile versions of the application, and maintain a single, reliable set of data.

In the case that a server connection is not available, which is not to be considered an unusual case for mobile phones, local storage is also used to store data yet to be synchronized with the server. A practical scenario for this event is when a user is out for a run and decides to log the activity, but the devices fail in establishing a connection with the server. The data are stored on-device and synchronization is reattempted when the application is next started.

3.6.2 Server

As presented in Figure 3.1 the client and server will have to communicate. The client will send requests to the server and the server will respond by performing some operation and communicating back the response to the client.

In order for the client and server to communicate the request and response format and parameters had to be agreed upon in order to understand each other. An API was created for this purpose. All the operations that are available to the client are described in the servers API discussed in Section 4.2.

In accordance with the MVC structure, the server is implemented as a controller that loads the libraries that it requires. The server has as described above several methods that perform different operations. The methods are using the libraries that were loaded in the controller but also load and use models to perform and process data from the database. The administrative interface uses the same structure and uses the same models but has a separate authentication library.

3.7 Platforms

The platforms specified were found to be too wide at the beginning due to our choice of development tool: Titanium Studio. It would allow us to create the application for mobile devices considerably easier than for desktop browsers. However, as the transition to PhoneGap was made the platforms supported were once again expanded, although the mobile side remained the primary focus.

The end result is compatible with both desktop and mobile, albeit in varying degrees. Certain functions do not translate directly from one environment to another, such as location sensing for the map feature, and notifications, which are built specifically for the mobile platforms.

The main focus was the Android platform as it was the easier mobile platform to develop for of the two main mobile platforms, since iOS requires proprietary hardware and software, including licenses. A majority of the authors are also in possession of android devices.

3.8 User interface

The principles that the iOS Human Interface Guidelines provide state that a splash screen is prefered to be avoided [11]. The Android GUI principles suggest the same, stating that they are “interruptions” and “not necessary” [12]. However, as the application does need some loading at the beginning, a splash screen is preferable over just an empty loading screen. The splash screen presents the logo of Gothenburg City.

As one of the main objectives was to digitize the physical card into an application, the UI should not differ too much compared to the original card. The original graphical profile was reused to make tiles, as described in Section 3.8.2.

At the beginning of the project the authors were not only struggling with trying to figure out a design that would fit the wide variety of users of the card campaign, but also designing the paper prototype into a digital form. The authors had limited knowledge of graphic applications leading to a steep learning curve to learn Adobe Indesign.

Some important aspects to consider are the differences in mobile and non-mobile devices. The devices utilize different forms of input. Mobile devices makes use of touch screens, accelerometers, compasses, and keyboards, while the older mobile devices may also use trackballs or D-pads[19] [13]. On the other hand, desktop computers and laptops generally incorporates only the keyboard, the mouse, and sometimes the track pad [13]. Different input methods have different strengths and weaknesses, which must be kept in mind when designing cross-platform applications.

3.8.1 Landing page

The first view that the user sees is the card view. The design for the card originated from the City of Gothenburg's initial graphics for the exercises in the physical card, shown in Figure 3.2. The colors and types of activities were kept, but additional tiles were introduced to access the added functionality in the application. The motivation for using the tiles in the application was their likeness to various popular UIs, perhaps most recognizable in the Metro UI of Windows 8, shown in Figure 3.3. The tiles are explained in more detail in Section 4.1.2.

3.8.2 Tiles

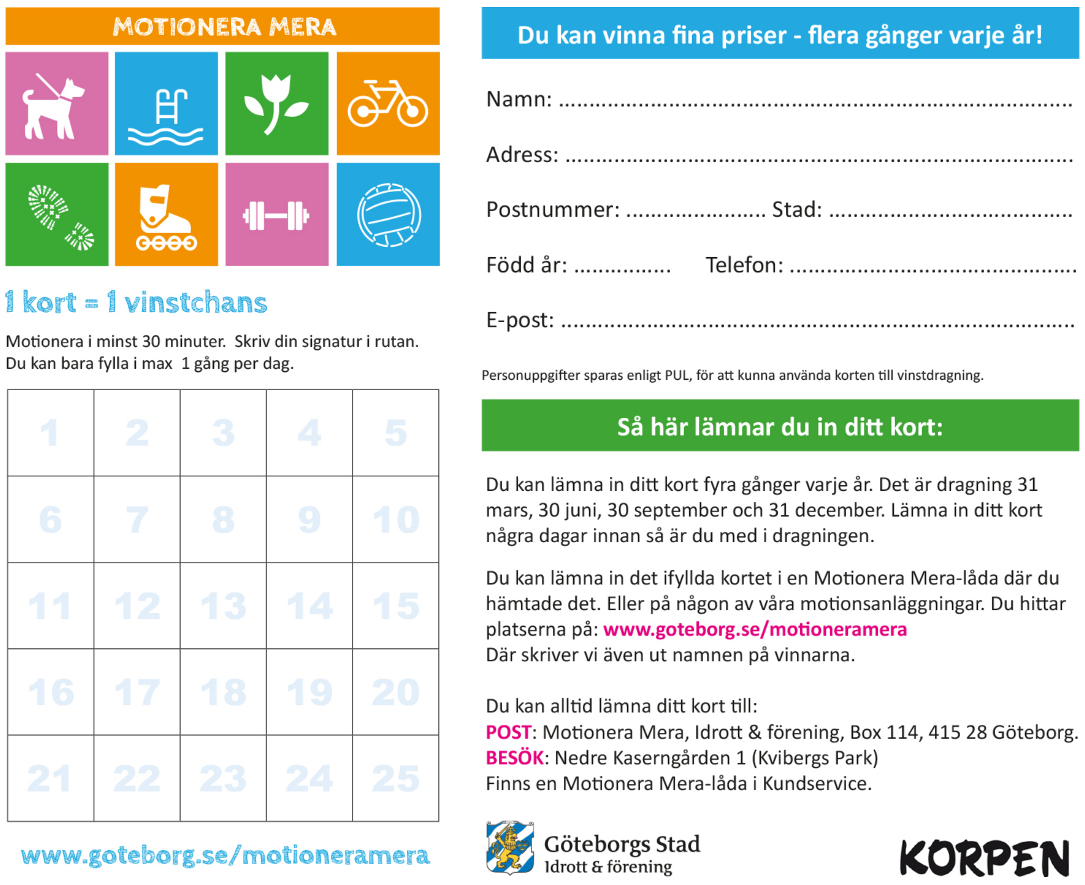
The tiles are the buttons which the user can interact with on the main page. There are two categories of tiles: the tiles to log activities, and the tiles to access other features such as the map and the news and announcements pages. The activity tiles are directly from the physical card itself, and were provided by the City of Gothenburg. The additional features required new tiles.

It was discovered early on that the additional tiles would have to follow the design of the activity tiles. Initially, the design followed the same color patterns as the graphical profile, but in order to avoid confusion the tiles were distinguished with a gray color instead.

3.8.3 Login screen

Some features of the application require that the user is logged in, such as the cloud-saving of user data. Since downloading the application requires a server connection, our decision was to make profile creation prominent in the startup of the application. The user could then create a profile in the same session as when downloading the application.

[19] A directional pad, usually with four directions leading to a plus sign shape.



MOTIONERA MERA

1 kort = 1 vinstchans

Motionera i minst 30 minuter. Skriv din signatur i rutan.
Du kan bara fylla i max 1 gång per dag.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

www.goteborg.se/motioneramera

Du kan vinna fina priser - flera gånger varje år!

Namn: ....

Adress: ....

Postnummer: .... Stad: ....

Född år: .... Telefon: ....

E-post: ....

Personuppgifter sparas enligt PUL, för att kunna använda korten till vinstdragning.

Så här lämnar du in ditt kort:

Du kan lämna in ditt kort fyra gånger varje år. Det är dragning 31 mars, 30 juni, 30 september och 31 december. Lämna in ditt kort några dagar innan så är du med i dragningen.

Du kan lämna in det ifyllda kortet i en Motionera Mera-låda där du hämtade det. Eller på någon av våra motionsanläggningar. Du hittar platserna på: www.goteborg.se/motioneramera
Där skriver vi även ut namnen på vinnarna.

Du kan alltid lämna ditt kort till:
POST: Motionera Mera, Idrott & förening, Box 114, 415 28 Göteborg.
BESÖK: Nedre Kaserngården 1 (Kvibergs Park)
Finns en Motionera Mera-låda i Kundservice.

Göteborgs Stad
Idrott & förening

KORPEN

Figure 3.2: The two sides of the physical card of the campaign. The back of the card is shown on the right.

The application does nevertheless function without a profile attached, and user data can always be synchronized with a profile at a later stage.

The login screen is displayed when the application determines that the user:

- has never logged in before;
- has had their session invalidated, i.e. the token has been rejected by the server;
- has cleared their cache.

The typical way of handling the process of logging in is to either provide an initial page only shown if the user is not logged in, or by displaying a popup. Popups are most frequently used on the desktop platform, and are less common on mobile phones. However, due to the structure of jQuery Mobile, the popup option was chosen.

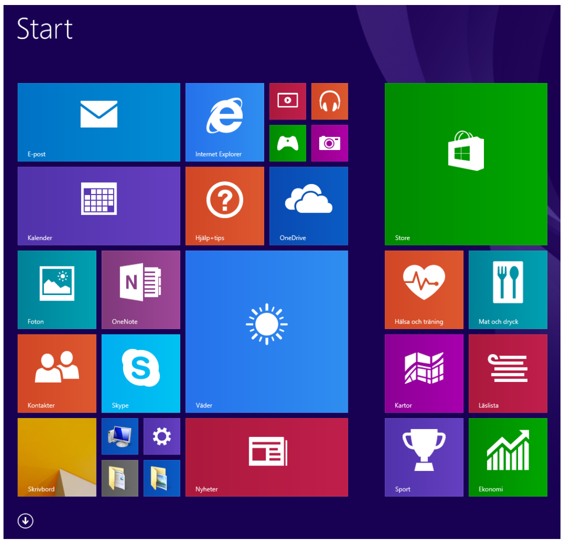


Figure 3.3: A screenshot of the UI of Microsoft's Windows 8. The squares and rectangles inspired the design for the landing page of the application.

jQuery Mobile uses the first page defined in the HTML document as the landing page. This means that there is no condition that can alter the initial path: the condition would be whether the user was logged in. If the login function were to be separated into its own page, the application would have to immediately navigate away. This gives a poor first impression of the application. The alternative would be to navigate from the initial login page to the card view, but since the login popup was designed to be shown rarely, this would effectively cause unnecessary delays each time the user starts the application.

The drawback of using a login page is that there must be a delay before it is displayed since, for instance, the WebKit[20] engine refreshes the page after loading it, causing the popup to disappear. Additionally, it would have been ideal if the user would not interact with any of the elements before the navigation, which is neither easy to guarantee nor a trivial case to consider with the frameworks used.

3.8.4 Scaling

The scaling of the UI was of particular importance as the application should look the same across different devices. We started adapting the interface to certain graphical profiles as a first solution, which meant that we had to categorize devices into classes based on the specifications of the devices. Google's Android provides such standard classes and is also a high authority in the field of mobile phones, which led us to follow

[20]`http://www.webkit.org/`

their standards. This meant adapting to the tall mobile phones and the wide tablet computers, desktop screens, and even TVs. Android makes use of the pixel density profiles; low, medium, high, and extra high, and also considers small, medium, and large screen sizes [14].

We found that some devices reported the pixel densities incorrectly, meaning the code would assume the wrong device profile for the devices. The pixel densities are the relation of the resolution, in pixels, of the device, and the physical screen area.

Density values of 0.75 are largely obsolete, while profiles of 1, 1.5, and 2.0 are common. Densities of 3.0 and greater are usually found in the flagship mobile phones as of 2014, such as the LG G2[21] or the Samsung Galaxy S5[22], with screen dimensions of 1080x1920 pixels and diagonal screen sizes of over 5 inches.

We later made use of a single set of larger image sizes which the application would scale. For example, the tiles of the main page are 512x512 pixels, even though the device displaying them might not have sufficiently many pixels to display them without scaling them down. The argument for using this method was that image loading is not a performance-heavy enough operation to justify the exponential workload of supporting both current devices and future devices; the application would eventually become unusable with this method. Another reason was that Google uses the same tactic for their Play Store; a page they should wish to work well with many devices.

For the activity view, as detailed later in Figure 4.6, a *contain* method was used to scale the image to the background of the screen. The method scales images such that the shortest dimension, i.e. width or height, is the limiting dimension, such that no part of the image is out of bounds of the window. Stretching images can make the image appear pixelated on high-resolution, large displays. This is mitigated by providing larger images.

The scaling of the tiles was implemented by the authors, as this provided better control of fine-tuning the interface. Such control includes the number of displayed tiles in either dimension, how large the margins should be between them, and how to position or align them. It can also be changed more easily to work in special scenarios, such as when the user changes from portrait view to landscape view by rotating their smartphone.

There are plugins available that solve similar problems, such as *hammer.js*[23] or *masonry*[24], but these add more functionality than required, such as animations, boxes of different sizes, or texts on the tiles. Hence, they lead to greater overhead and lower

[21]`http://www.lg.com/global/g2`
[22]`http://www.samsung.com/global/microsite/galaxys5/`
[23]`http://eightmedia.github.io/hammer.js/`
[24]`http://masonry.desandro.com/`

performance.

3.9 Performance and optimization

In order to achieve greater performance, page content was generated partly with JavaScript code, such as in the news and announcements page. Generating the HTML content means that the time it takes for the application to start is decreased, as there are fewer elements to load: elements are only loaded when they need to be shown. This minimizes the number of elements, consequently reducing query times and memory usage. In short, generating page content allows for faster startup of the application in addition to lower memory usage and faster queries.

Some performance-heavy operations have been avoided, such as overusing shadows on elements, or images with transparencies. If the background behind the image is static, then the image can be optimized by filling the transparent sections with that background color instead. Another example of a performance-heavy feature is animation, particularly on older devices. Animations during page transitions also result in the application feeling sluggish. Further discussion regarding this topic can be found in Section 2.4.3.

There are also several aspects of server-side optimizations that were important to consider when developing the server.

The database communication is using carefully constructed database queries that utilize as much of the built in technologies in MySQL as possible, which is preferable over doing the processing in PHP. Additionally, much of the data requested by the users are not unique and it is possible to cache the queries and their corresponding resulting data. This reduces the amount of queries and reduces the load on the database as data are instead read from the cache on the file system. The Codeigniter framework has support for query caching but also allows for loading models and libraries when they are needed so that only the code that is required to run a particular method is loaded.

The administrative interface generates HTML markup which can be compressed in order to reduce the data that has to be transferred. The same is possible for the CSS and JavaScript files. Such compression is common and is referred to as *minification.*

3.10 Security and authorization

In securing the server and the user data, several aspects had to be taken into consideration. The server therefore has several security mechanisms and levels of security put into place.

The token authentication system discussed in Section 3.10.1 is the authentication system that authorizes clients to perform operations on users data. The administrative interface is instead using *session variables*[25]. Additionally the passwords are protected so that if the database is compromised the passwords will remain difficult to obtain as discussed in Section 3.10.4.

3.10.1 Token authentication

All requests to the server that retrieve and update any information regarding the user are protected by a user authorization system. In order for a client to change or retrieve any information related to a specific user, a token must be attained. A token is in essence a key that is generated by the authorization system that grant the client permissions to run certain operations on the server. This key is stored as a tuple in the database containing the key itself, a user identifier index and an expiration date.

The client is allowed to query the server whether the key has expired. After it expires the user will have to use their login credentials in order to generate a new token. The token is however kept alive when the user is active by adding a time-to-live (TTL) from the present time at any point the server authorizes an operation with that token.

Tokens are generated using the *openssl_random_pseudo_bytes* function of the OpenSSL library. The tokens are then encoded with the *base64* function and converted from binary data into 7-bit ASCII characters. They are only stored in the database if the generated token is unique.

3.10.2 OpenSSL

Since the server is using OpenSSL to generate *salts*[26] and tokens, it is important to take the OpenSSL Heartbleed bug into consideration. The versions of OpenSSL vulnerable to attacks following the heartbleed bug are OpenSSL 1.0.1 through 1.0.1f [15]. This requires the system hosting the server not to run the affected versions of OpenSSL.

[25]Session variables are variables that are stored on the file system, meaning that they are not cleared when the web page is refreshed.

[26]A salt is a unique string appended to the password before it is encrypted using a one-way encryption algorithm, also known as hashing

3.10.3 Authorization

In order to create a user, a username and password must be provided. The user is then stored in the database as a tuple containing a user identifier, username, password and a salt. The password is simply an encrypted string based on the password provided by the user and the random salt generated by the authorization system.

When a user tries to authenticate using their username and password, the authorization system will encrypt the password using an encryption algorithm and the salt in the user record associated with the provided username. The value of the encrypted string will then be compared to the password stored in the database and, if they match, a token with a TTL will be generated and returned to the user. Whenever a user already has attained a token it can simply query the server with that token and the server will check that the token is alive in which case it will perform the action, otherwise the user has to reauthenticate.

3.10.4 Password security

If the passwords were to be compromised it is preferable to have them encrypted using an algorithm that is computationally expensive forcing the attacker to spend more time decrypting the passwords. The downside is that it will also require more computation for the server to encrypt and compare passwords as is described in Section 3.10.3. Blowfish was chosen as the algorithm for encrypting passwords since it is efficiently implemented in PHP and there is no efficient *cryptanalysis*[27] [16]. There exist weak keys in Blowfish [17] but it is still a widely used algorithm that is viable for the purpose of this project.

Additionally, the passwords are salted for essentially the same purpose: to make it computationally expensive for the attacker. If the database has been compromised it should be expensive to attain a password without compromising the performance of the system. A common method for cracking passwords is to use or generate rainbow tables, which are dictionaries or mappings from strings to hashes. An attacker will generate all possible passwords of some length and encrypt those, comparing the output against the values in the database. If a unique string is provided for each user and appended to the password before encryption, this task will become very computationally expensive. The attacker will have to generate such a table for each individual user, e.g. by brute force. The security or secrecy of salts is thus not important, only that they are unique.

[27]Cryptanalysis describes ways of exploiting the cryptographic algorithm in order to decrypt the plain text without knowing the key.

3.10.5 Synchronization

As described in Section 3.6.1, the local storage must be synchronized with the server. Synchronization is done by having a unique time stamp assigned to each activity created. When a set of activities is sent to the server, it inserts the time stamps and the associated data only if the time stamp is not found in the database. After sending the data and receiving a confirmation, the client can omit its local data and replace it with the data on the remote server.

Another synchronization problem occurs since queries are running concurrently: the server must ensure that there are no queries that need to be executed atomically[28] but are not. The server protects against this by creating a type of lock using transactions. The server will create a lock that blocks any other queries than the ones specified inside of the lock from executing. When the code and queries inside of the lock complete their execution the server releases the lock.

3.10.6 Input validation

The application will not necessarily filter the user input and the server could also be directly queried sending HTTP post requests. It is thus important to ensure that the data are properly validated and filtered to prevent unwanted data to be stored in the database.

3.10.7 Database constraints

The server scripts protect against and filter any input data in order to not break the structure of the database. However, it also has constraints that ensure that no such input will be stored. As an example of this, no two users may have the same username. Therefore it has a constraint that enforces usernames to be unique. In the database there are several constraints that protect against invalid input data.

3.10.8 SQL injections

In order to retrieve or store data to the database one uses queries that are written in a variant of the SQL language to specify what data one wishes to retrieve or store. It is

[28]Atomically means that, given a set of queries, they are either all executed without interruption, or none of them are executed.

often required to create SQL queries based on user input, for example in order to store user profile information. SQL injections are ways of manipulating the user input in order to modify the SQL queries and have it execute potentially malicious code.

In order to protect a system from SQL injections, the inputted data is filtered so that it does not contain any unexpected characters that could potentially modify the SQL queries. In CodeIgniter, the database queries are constructed using a database library which is set up to automatically filter the input data. The filtering protects against attackers attempting to inject SQL as no injected SQL is executed but rather interpreted as a string.

4 Results

This chapter provides illustrations of the achieved features, and describes the end result of both the client side and the server side of the application. Most of the client-side results are presented with screen shots of a prototype for the application, as that presents the results from the perspective of the end user. This chapter also presents results in both the design of the application and the security aspect. As client-side performance has not been measured, it is not presented.

The graphical components shown in the figures have been created solely by the authors, with the exception of the Gothenburg City logo and the activity tiles of the physical card, as noted in Section 3.8.2. Frameworks have provided a basis for the GUI. The screenshots were taken with an Android smartphone.

The implementation process and explanations of the terms found in this chapter are found in Chapter 3.

4.1 Graphical user interface

The GUI of the final application was of importance, since it was heavily involved with the requirements set by the client, as detailed in Section 2.2. This section presents some of the most important aspects of the GUI.

4.1.1 Splash screen

The splash screen, or *loading screen*, as shown in Figure 4.1, is the first screen that greets the user when starting the application. As explained in Section 2.1, the Gothenburg City logo should be shown, and the authors suggested the splash screen as an unobtrusive option. The splash screen is only shown while the application is loading, hence there is no added delay. The load time, during which the splash screen is shown, is generally no longer than a few seconds, and it may go unnoticed on faster devices.

Figure 4.1: The splash screen shown during startup. The image consists of Gothenburg City's official logo, provided by Gothenburg City themselves.

4.1.2 Card view

The card view is the main page, or landing page, of the application, and is shown in Figure 4.2. It is the root of the navigation tree, containing eight activity buttons and four navigation buttons. These buttons are the tiles discussed in Section 3.8.2, and together they comprise the entirety of the set of interactive elements on the landing page.

The user may use the navigation tiles to access the added features summarized in Section 2.1. As seen in Figure 4.2, the additional four tiles are colored gray to distinguish them from the activity tiles.

At the bottom of Figure 4.2 is a progress bar which indicates how many of the 25 exercises, on unique days, that the user has saved.

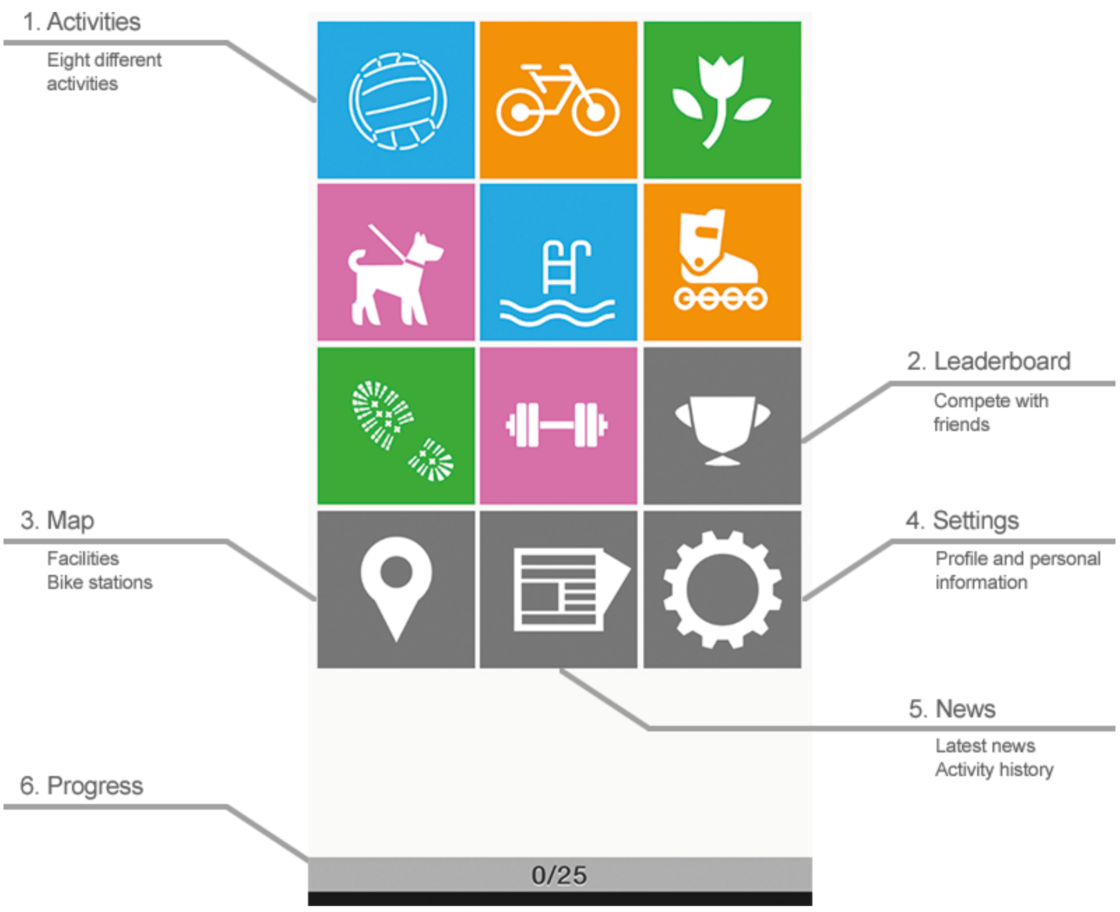
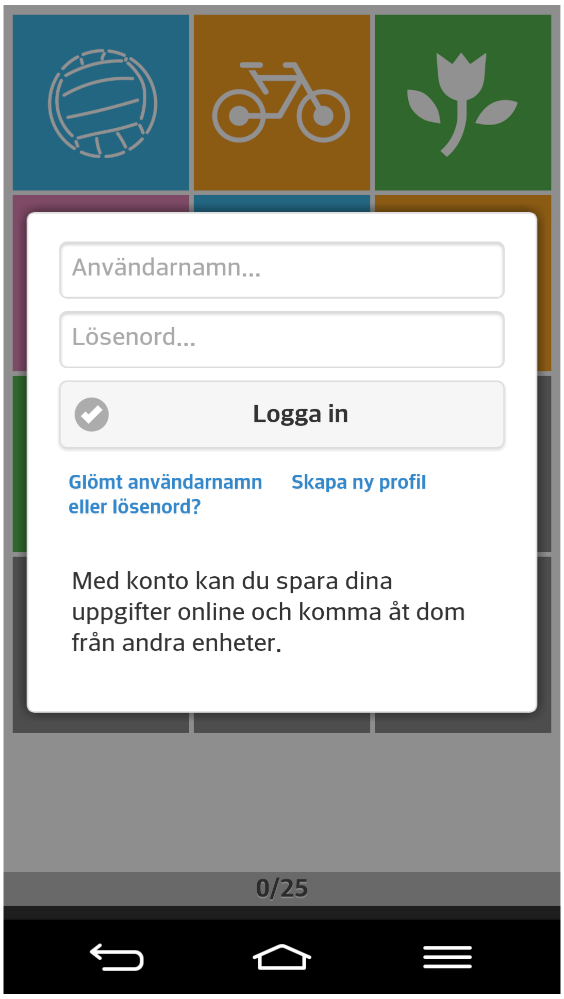


Figure 4.2: The card page explained. The figure separates the colored tiles (1) from the gray ones (2) to give a visual clue that the gray buttons are not activities, and briefly explains the five classes of tiles. The activity tiles (1) are part of the physical card in the card campaign of the City of Gothenburg. The figure shows the tiles of the additional features (2), (3), (4), and (5). (6) shows the progress bar indicating the number of logged activities counted toward the full card of the campaign.

4.1.3 Login

When a user starts the application for the first time, a login dialog is shown. The dialog can be dismissed by pressing the area outside of it. As long as the user has not logged in, the dialog will be available under the settings page, as shown in Figure 4.11b.

If the user wins a prize in the promotion they must register their shipment information. Although it is possible to provide the information when registering, it is not required at that point. The reason is that it would be a burden to the users who are either not interested in the promotion or have not yet won.

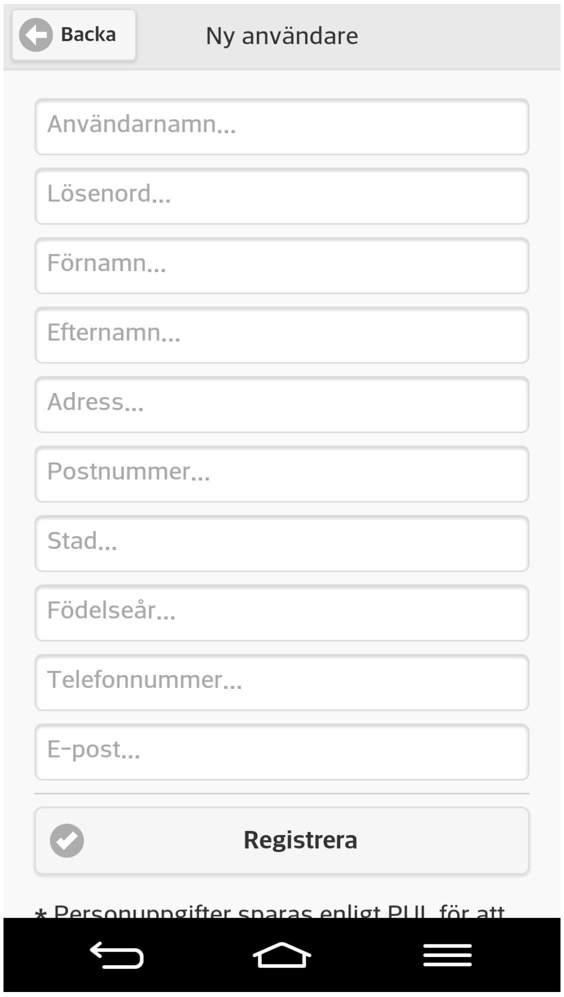


Med konto kan du spara dina uppgifter online och komma åt dom från andra enheter.

Figure 4.3: The login dialog covering the main page.

4.1.4 Register

The registration page allows the user to register a profile by providing at least a username and a password. The other fields, shown in Figure 4.4, are optional.



* Personuppgifter sparas enligt PUL för att

Figure 4.4: The registration screen, accessed via the login dialog.

4.1.5 Profile recovery

If a user forgets their password, the profile recovery page allows them to reset their password. The user must have signed up with an e-mail address which they can access. When submitting a properly formatted e-mail address to the recovery data field, an e-mail with a password reset link is sent to the e-mail provided, if there is a profile tied to the e-mail in the database. The recovery page is shown in Figure 4.5.

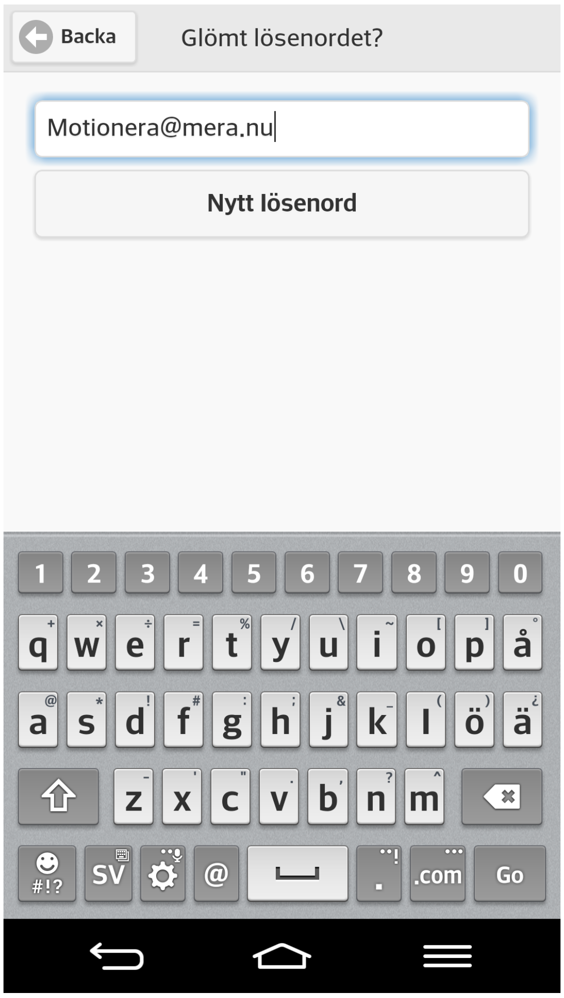


Figure 4.5: The password recovery page, accessed via the login dialog. The user may enter their email address to be sent details regarding resetting their password.

4.1.6 Activity logging

If one of the activity buttons of the card page is pressed, the application navigates to a screen in which the user may log an activity, as shown in Figure 4.6. The user is required to select a time from the two spinning wheels, as previously mentioned in Section 3.5.3. By design, the wheels do not accept a duration under 30 minutes, as exercises under 30 minutes do not lead to a mark for the promotion. In addition to selecting the duration, the user can optionally add notes connected to the exercise.

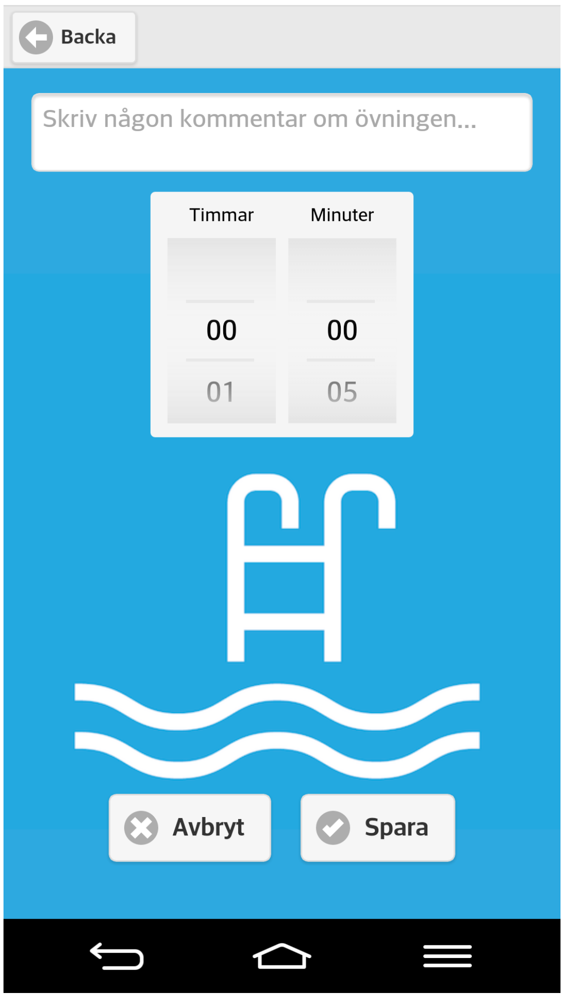


Figure 4.6: The screen that allows the user to save an activity. The background of the page depends on the activity that was pressed in the card page.

4.1.7 Map

The map page shows Gothenburg with all of the client's facilities marked on the map. There is also the possibility to filter between facilities and bicycle stands (Styr & Ställ). An example is presented in Figure 4.7.

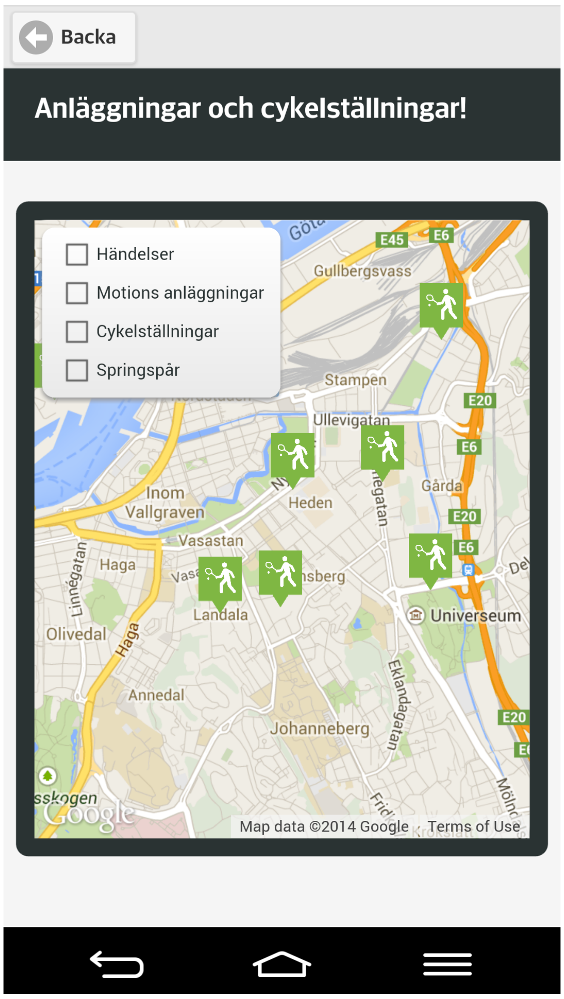
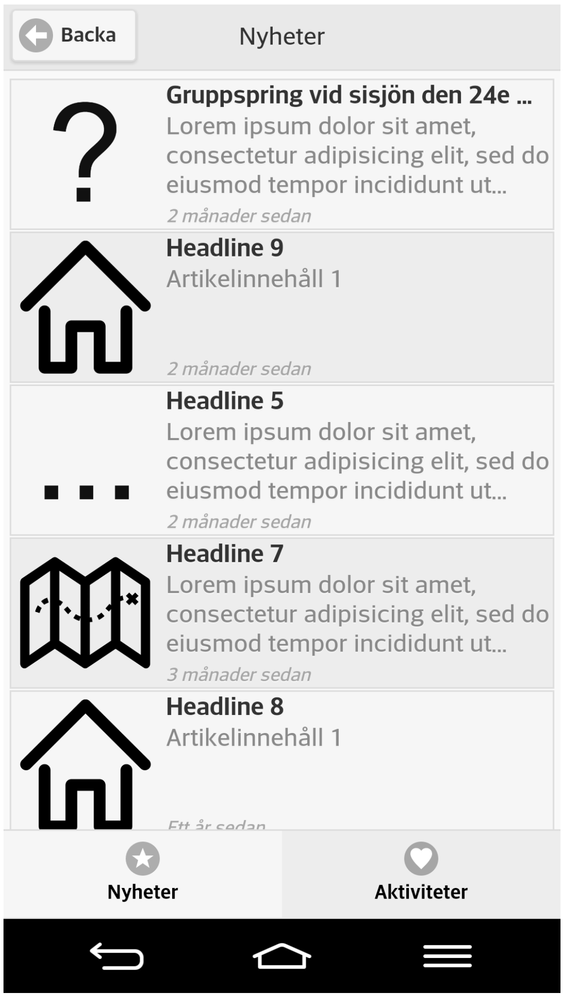


Figure 4.7: Displays the exercise facilities and "Styr & Ställ" bicycle stands of the City of Gothenburg. The figure also includes events and running tracks which are not implemented.

4.1.8 News

The news page displays the announcements that the City of Gothenburg are able to write through the administrative interface, as mentioned in Section 4.3. The three components are the headline, the main text area, and the time stamp. As described in Section 3.5.5, the news section uses time stamps which update in real time. The news page is shown in Figure 4.8.



Gruppspring vid sisjön den 24e ...
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut...
2 månader sedan

Headline 9
Artikelinnehåll 1
2 månader sedan

Headline 5
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut...
2 månader sedan

Headline 7
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut...
3 månader sedan

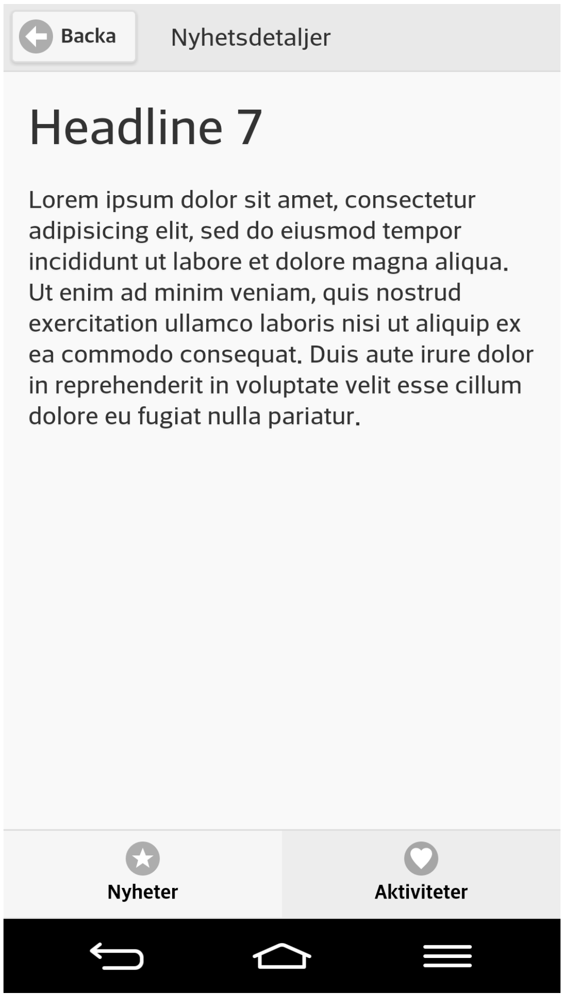
Headline 8
Artikelinnehåll 1
Ett år sedan

Figure 4.8: The news and announcements view.

As shown in Figure 4.8, the page has a tab bar which allows the user to navigate to the activity history view, as described in Section 4.1.9.

News item details

Although the content of a news post is previewed in the news and announcements page, as shown in Figure 4.8, there may be news posts which do not fit in the preview. The user may select any news item to be displayed in a new page, an example of which is shown in Figure 4.9.



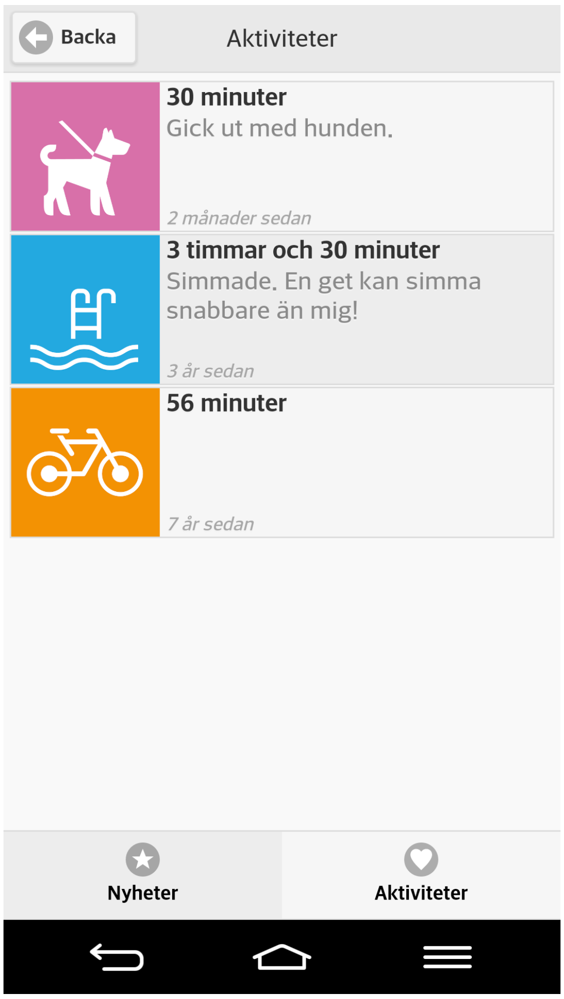
Headline 7

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Figure 4.9: Shows a single news post in full-screen. The area is scrollable and provides a large amount of space for more content in each news post.

4.1.9 Saved activities

The activities that the users save are synchronized to their profile if they are logged in. Whether the users are logged in or not, their saved activities can be viewed. The activity history page is shown in Figure 4.10.



Backa

Aktiviteter

30 minuter
Gick ut med hunden.
2 månader sedan

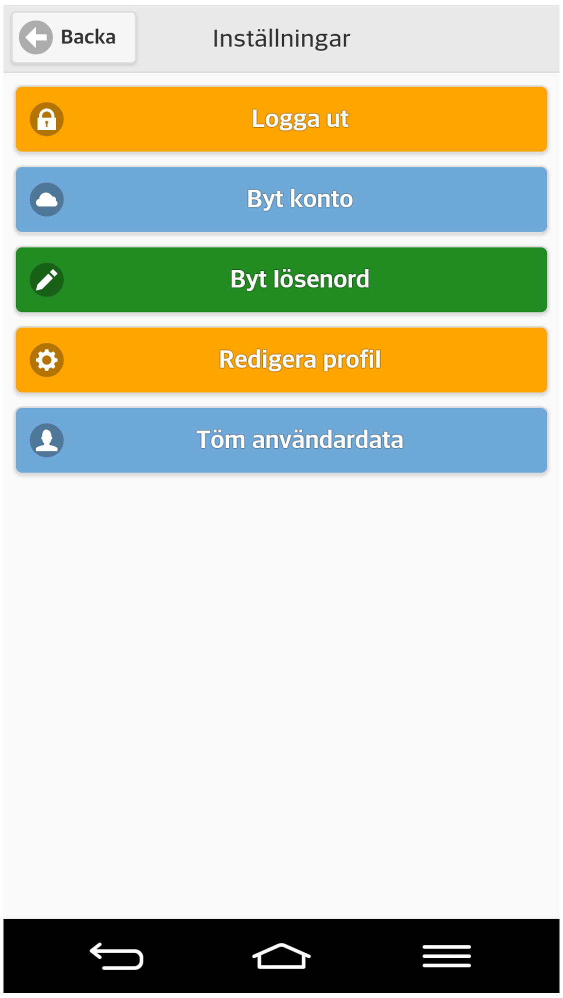
3 timmar och 30 minuter
Simmade. En get kan simma snabbare än mig!
3 år sedan

56 minuter
7 år sedan

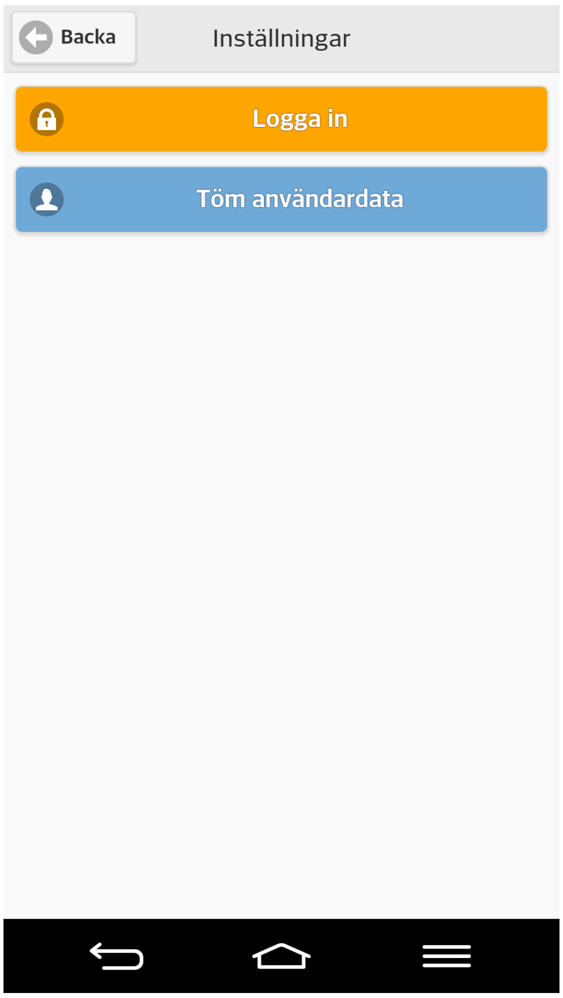
Nyheter

Aktiviteter

Figure 4.10: Displays a list of the activities that the user has saved.



(a) The user is logged in.



(b) The user is *not* logged in.

Figure 4.11: The settings pages. The options available depend on whether the user is logged in.

4.1.10 Settings

The user finds the options for managing profiles: logging out, changing profile, changing password, editing their profile, or clearing their user data. Clearing the data results in removing any locally stored user data, such as unsynchronized activity logs. The settings page shown when the user is logged in is presented in Figure 4.11a, while Figure 4.11b presents the options available when the user is not logged in.

4.1.11 Notifications

Notifications are provided when appropriate to assist the user in following what the application is doing. A special case where notifications are essential are when the client attempts to connect to the server and fails. The user must then be made aware of this event. The notification box implemented in this application is shown in Figure 4.12. The visuals of the notification box are largely inspired by the default notifications of the Android operating system, which are known as *toasts*[1].

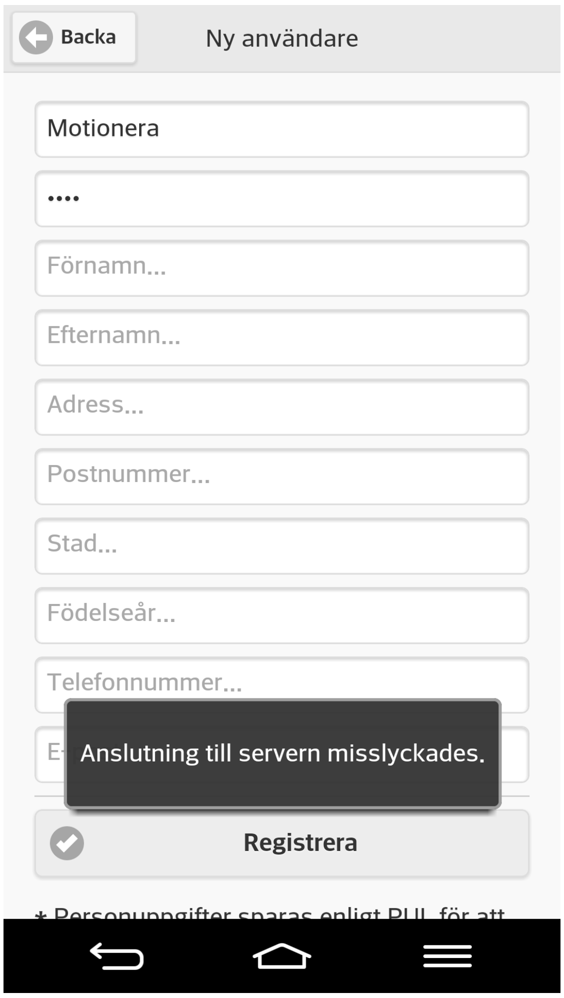

Motionera
••••
Anslutning till servern misslyckades.

Figure 4.12: A notification is shown to the user noting that the registration failed due to failure to establish a connection with the server. Notifications are used to display information unobtrusively to the user in various scenarios where the user needs to be aware of the actions of the application.

[1]`http://developer.android.com/guide/topics/ui/notifiers/toasts.html`

4.2 Server

The client communicates with the server according to the API that was created. The API has support for essentially five classes of operations: authentication, user profiles, activities, news and map. The different features of the API are described further in Table 4.1.

Feature	Description
Authorization	Enable users to login and the client to test if they are still logged in.
Users	Enable the client to register user profiles, reset password and logged in users to update their user profile, change password, add and remove friends.
Activities	Enable logged in users to register, update, retrieve and remove exercise activities.
Map	Enable the client to receive location geocodes and their corresponding metadata.
News	Enable clients to receive news headlines and article content, which is published using the administrative interface 4.3.

Table 4.1: Summary of server-side features.

4.3 Administrative interface

An administrative interface was constructed in which the system administrator can create and manage news articles, users and their information, and facilities as summarized in Table 4.2.

Feature	Description
News publishing	View, edit, add, and delete news and announcements.
User administration	View, edit, add, and delete user profiles.
Manage the map	View, edit, add, and delete locations on a map that shows all of the currently added locations.
Usage statistics	View some key statistics about the users and their usage.

Table 4.2: The features of the administrative interface.

5

Discussion

In this chapter, we discuss some questions that were prompted by the development process, and reflect back on the choices that we made. We also describe the limitations of the application and explain particular cases where the end result was not what was anticipated. We begin by relating mainly to the contents of Chapter 3: the planning, the development environment, and the process. After that we expand on unforeseen circumstances, limitations with the prototype, and reflections on the choice of development tools and their impact on the end result.

5.1 Planning

The planning was made with a larger set of features than we expected to finish, in order to add more substance to the project. The main feature to implement was the digitization of the physical card. In retrospect, the card function could have been implemented more thoroughly had the scope not been widened. Whereas it would have led to more time for user testing, for instance, it would also have resulted in an application with limited functionality. It would also not have been as challenging as a project. We and the City of Gothenburg both wanted more than the card system implemented in the prototype.

While the requirements specified by the client were fulfilled, we were not able to put our efforts into making another large-scale feature such as a leaderboard system. The results do, however, show a working prototype with both client and server sides handling the promotion. We did not achieve some of the planned results, in part because the promotion card feature took longer than anticipated to implement.

An early version was developed specifically for one of our devices. At that point, the GUI was hard-coded for a particular device. The biggest challenge was adapting the application to other devices, mostly due to the scaling of the user interface, as discussed in Section 3.8.4.

A significant task in the planning of the project was to create the core of the application around which other features would be built. Due to the nature of web languages, the core had to be continuously reworked. For instance, we learned that generating HTML code by JavaScript was more efficient, so existing HTML code was converted to JavaScript generation instead. Similarly, the web languages do not inherently allow for the MVC model which makes the coding more difficult since we are used to object-oriented programming. In short, the designing of the system core meant that features that were to be integrated were somewhat slowed down.

Relating to the milestones of Section 3.2, what was eventually accomplished was the first milestone and one of two features of the third. We implemented the card promotion and added partial GPS functions. However, the map feature was redesigned to show both facilities and the bicycle stands of “Styr & Ställ.” The priority of features changed during the project, both as we learned more about the implementation languages, but also as the progress gave us a better picture of the entire project. Among the initially planned features, some either saw no progress or were continually postponed, as discussed in Section 5.4.

We learned that cross-platform development can make a relatively easy task more demanding, especially when developing for both desktop computers and mobile phones. Some of the basic features largely did not require adapting the code to different platforms. For example, the difference in how browsers interpret the code relating to styling of the GUI is minimal.

5.2 Development environment and platforms

One of the main challenges was our lack of familiarity with the different coding languages and the tools used. Most of us have not created web applications before. Since cross-platform implementation was an important goal, another challenge was the partial unfamiliarity with other operating systems. With only one of the authors having access to iOS hardware, the implementation toward that platform in particular became minimal. On the other hand, the rest of us were familiar with the Android system and could offer our own experiences with what is expected in Android applications. Some features are unique to mobile devices, for example that the display rotates when the device is turned.

As development was made with web languages, any browser could interpret the code. This meant that the testing was continuously done while developing on the primary browsers and devices used, in this case the Android platform and the Chrome and Firefox browsers. The application may therefore show bias toward these platforms.

5.3 Unforeseen outcomes

When Titanium Studio was initially chosen as the main cross-platform tool as opposed to PhoneGap, it was easy to build the code conditionally depending on the platform used. An example is the navigation bar which is required on iOS devices since iOS lacks a built-in back button. On Android devices, the back button is provided either by a hardware button or by a panel of buttons in software on the screen of the device itself. Similarly, back buttons are implemented in software in desktop browsers and as hardware buttons on desktop mice. This makes ensuring that navigation works on iOS an unexpected issue. With PhoneGap, we did not realize a way of conditionally inserting a navigation bar only when it is needed, so the navigation bar was integrated with the application for the builds of each platform. The navigation bar was not a major problem, as vertical space was sufficient. The bar also introduced a label for each page which would assist the user in knowing where in the application they were.

Another unexpected, but mostly minor difficulty arising from the chosen tools was the general unreliability in how the interpreters of different browsers handle the code. The elements could be wrongly positioned or displayed in a way not truthful to the code written. Most of these issues were found to originate from the jQuery Mobile framework. The framework changed the design in various ways, such as the exclusion of a checkbox on the login screen. The reason was that it caused the page to flash when selected. The state of the checkbox, that is, whether it was checked or not, would have decided whether user details would have been saved once inserted. The combination of many such small issues had an impact on the final design of the application.

We were also hoping to be able to implement a calendar since it was originally included in the first milestone. We found a suitable plugin for the map that was estimated to take less time than we thought during our planning phase. After assessing the priority of the calendar and map, we realized that the map had a higher priority since it was a request by the City of Gothenburg, which lead to prioritizing the development of the map function instead.

5.4 Limitations

The activity logs were originally designed so that the durations on each day would be summed up to check whether the threshold of 30 minutes was reached. Currently, the application only checks for single activities. This makes logging activities under 30 minutes not affect the progress of filling the card.

Although the map feature is aware of the location of the user by initially centering the view on them, the GUI does not indicate where they are. This may lead to disorientation should the user pan or zoom in on the map.

We would also have liked to implement the leaderboard system. As it currently stands, it is only a sketch. Although some functionality is implemented server-side on the database, mainly in the connections of user profiles through a friends system, the development of the leaderboard feature was not started for the prototype.

User testing was also part of the original plan. We asked the City of Gothenburg if they could provide assistance, but they lacked the capacity to do so. We also found that time invested in implementation would be more rewarding instead of focusing on user testing. Even though the application was tested on various browsers on desktops and on multiple mobile devices, the application would benefit from additional, thorough testing before it can be regarded as complete.

An aspect which is affected by other limitations is navigation in the application. The navigation was designed relying on the implementation of the planned features, thereby leading to gaps in the GUI. As an example, the calendar would have been grouped in a tabbed page with both the training schedule and the list of activities logged. Instead, the activity list is grouped in the tabbed page with the news and announcements screen.

5.5 Reflections

As the core goal of digitizing the card was expected to be reached well before the end of the project, additional features were planned. Adding more substance to the project was made simpler by the choice of making a web application, since a significant portion of the code would be compatible with all of the target platforms.

The prototype did not include enough of the planned features to make the application feel complete. Had the project group been larger, there would have been more manpower to allow for dividing the group to work specifically on different platforms.

An example of an early adaption to a native platform is the Android menu, partially implemented in the Android-specific build of the application. The native code consists of Java, but could also send JavaScript code from the native environment. The Android menu had lower priority than other tasks, and was not integrated.

While we chose to use PhoneGap as the main cross-platform tool there were other alternatives, such as Titanium Studio and Xamarin. Titanium Studio was error-prone, making development difficult. As for Xamarin, there were additional costs for organizational usage, thus the City of Gothenburg would have to submit it to App Store and Google Play, which they did not have a budget for. This made it difficult to consider the alternatives to PhoneGap as viable options.

5.5.1 Change of goals

If we were allowed to change the purpose of the application, or rather if we were simply requested to create an unspecified exercise application with free reign, the resulting application would likely have become more of a training application in line with the currently most popular health and fitness applications rather than a casual exercise application. Planning would have gone more smoothly as there would have been fewer constraints and more space for innovation.

Limiting the scope to only one platform, either iOS or Android, would help focus the process as well as the application as the developers could work together more closely. It would be easier to implement and test since there would only be one code base. It would also have likely produced a prototype with higher quality and fewer bugs.

5.5.2 Change of method

Scrum started well, but after only a few weeks we started diverging from it. The problem was partly that we were interrupted by the planning report once we started development, but also because Scrum works better when working every day and for longer periods of time than two to four hours. Scrum might have also worked better if it was physically implemented with post-its on a white board, as there would be no need to log in back-and-forth to check on what had been done and what needed to be done. One would visually and directly see a greater summary of what tasks needed to be completed.

The Scrum meetings were still useful in that it helped to keep us up to date with the progress of the project. It also made it clear when we were behind schedule, and made it easier to learn about the issues the others in the group were having.

5.6 Designing for native

One thing we had to consider was how to make our application feel like a native application. There have been some complaints from the PhoneGap development community that their applications have been rejected from Apple's App Store [18]. The main reason seems to have been that the application did not feel native enough, which means the feeling of a native coded application compared to one with web language.

Apple provides clear guidelines and principles for designing UIs for their devices. iOS 7 embodies deference, clarity and depth as their main subjects [11]. Android's UI guidelines focus on *Enchant me*, *Simplify my life* and *Make me amazing,* which are topics that are not as clear as what Apple have defined [12].

It seems that Apple is stricter in letting applications into the App Store than Google Play, some amount of effort has been into making the application as native as possible.

5.7 Interaction with the City of Gothenburg

During our meetings with the City of Gothenburg we showed them what we had been working on. We also received feedback and explained the plans we had regarding future development. One of the ideas that they were especially pleased with was the statistical usage data that the application could potentially gather. They had not thought of that possibility earlier, which is not surprising as it is a relatively new way of gathering exercise data as mentioned in Section 1.1.

Often when engineering students communicate among each other, they tend to use similar terminology, since they have similar backgrounds. It was a valuable experience for us to work with people with other backgrounds.

5.8 Future work

The application was developed with more features planned than implemented. This means that the navigation is not as planned: in order to access the activity history, for instance, the user would select the news tile, which is not as intended. With the planned features complete, the flow of the application would make more sense as the training schedule, calendar, and activity log history would be grouped in the tabbed view, with the news and announcements feature having its own full page.

The primary example of incomplete features is the leaderboard view, which lacks functionality but remains present in the prototype. Implementing the leaderboard feature would also justify the work put into the user profiles. It would also add substance to the currently implemented features, as the leaderboard could work in tandem with the news feature by allowing for group events among users that are friends. In addition, the user statistics feature would also have served as a stepping stone toward the leaderboard feature.

We also had plans to implement social functions such as profiles and to be able to synchronize with Facebook or Twitter. Similarly, the possibility of taking pictures was initially discussed with the client. These would also be connected to the leaderboard feature in that the user profile picture could be taken from the same page. Another possibility would have been to be able to upload images so that other users can view them on the map.

As the application gathers a lot of data, there is a possibility of reusing the data in user statistics, which was of interest for the City of Gothenburg. One simple use for the data are that if they find that the most common activity is running, then they could host more running events to get more people active.

There are also some events which are initially overlooked when designing an application, such as the user receiving a phone call while using the application. These events can be controlled natively on Android by executing specific code during these events, such as when the application becomes paused or resumed.

Another aspect to take into consideration is that the user data are currently sent using an unencrypted communication. In order to further enhance security, the user data should use an SSL-encrypted connection, which would prevent eavesdropping.

6 Conclusions

The goal of the project was to deliver a prototype application to the City of Gothenburg. The application was required to implement the functions of their currently active card campaign. The application was also required to contain a map showing the locations of their facilities. As the prototype incorporates the requested features, the main goal of the project was achieved. While there are still small adjustments to make, mostly related to integrating the additional features, the application is functional for its purpose.

The application provides the possibility of publishing news and announcements. The feature is supported by an administrative interface which allows for creating, editing, and removing news posts.

The map showing the exercise facilities of the City of Gothenburg was one of the two primary goals of the project. The feature was implemented and extended by including the “Styr & Ställ” bicycle service, which the City of Gothenburg also administrates. The map feature uses the Google Maps API meaning that the user can interact with the map by panning and zooming instead of viewing a static image of a map.

The application was required to store user data in order to track the progress the user has made toward a full card. Our solution was to store the data on a server and keep the latest copy on-device. This means that the user has offline access to their data.

Another advantage of storing user data server-side is the synchronization feature that allows users to keep multiple devices up to date. To store user data server-side, user profiles were implemented. Encrypted passwords and a token authorization system are used to ensure the privacy of users.

Since the application was built relying on web technologies, it can be used on any device with a web browser. User data can therefore be managed through multiple devices.

Finally, the graphical user interface scales to the device the application is running on. It is responsive to resizing the application window and adapts to orientation changes on mobile devices.

Bibliography

[1] J. Bort-Roig, N. Gilson, A. Puig-Ribera, R. Contreras, S. Trost, Measuring and Influencing Physical Activity with Smartphone Technology: A Systematic Review, Sports Medicine 44 (5) (2014) 671–686.
URL http://dx.doi.org/10.1007/s40279-014-0142-5

[2] PUL, Statliga myndigheters behandling av personuppgifter, [Online; accessed 2014-05-18] (2014).
URL http://www.datainspektionen.se/lagar-och-regler/personuppgiftslagen/e-forvaltning/statliga-myndigheters-behandling-av-personuppgifter/

[3] Regeringskansliet / Lagrummet, Personuppgiftslag (1998:204), [Online; accessed 2014-05-18] (2010).
URL http://www.riksdagen.se/sv/Dokument-Lagar/Lagar/Svenskforfattningssamling/Personuppgiftslag-1998204_sfs-1998-204/?bet=1998:204

[4] E. Marcotte, Responsive Web Design, [Online; accessed 2014-05-15] (2010).
URL http://alistapart.com/article/responsive-web-design

[5] Google, Android NDK, [Online; accessed 2014-06-05] (2014).
URL https://developer.android.com/tools/sdk/ndk/index.html

[6] S. Diwakar, Titanium vs Phonegap vs Native application development, [Online; accessed 2014-05-11] (2012).
URL http://www.sapandiwakar.in/api-research-study-iphone-and-android-applications/

[7] jQuery Foundation, jQuery: The write less, do more, JavaScript library, [Online; accessed 2014-05-04] (2014).
URL http://jquery.com/

[8] MIT, The MIT License (MIT), [Online; accessed 2014-05-18] (2014).
URL `http://opensource.org/licenses/MIT`

[9] M. Dellanoce, Fast Touch Event Handling: Eliminate Click Delay, [Online; accessed 2014-05-04] (2014).
URL `http://phonegap-tips.com/articles/fast-touch-event-handling-eliminate-click-delay.html`

[10] R. Fioravanti, Creating Fast Buttons for Mobile Web Applications, [Online; accessed 2014-05-04] (2014).
URL `https://developers.google.com/mobile/articles/fast_buttons`

[11] Apple, Designing for iOS 7, [Online; accessed 2014-05-10] (2014).
URL `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html`

[12] Google, Design principles, [Online; accessed 2014-05-10] (2014).
URL `http://developer.android.com/design/get-started/principles.html`

[13] R. Ghatol, Y. Patel, Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5, Apress, 2012.

[14] Google, Supporting Multiple Screens, [Online; accessed 2014-05-04] (2014).
URL `http://developer.android.com/design/get-started/principles.html`

[15] OpenSSL Security Advisory, TLS heartbeat read overrun (CVE-2014-0160) (2014).
URL `https://www.openssl.org/news/secadv_20140407.txt`

[16] T. Nie, T. Zhang, A study of DES and Blowfish encryption algorithm, in: TENCON 2009-2009 IEEE Region 10 Conference, IEEE, 2009, pp. 1–4.

[17] O. Kara, C. Manap, A new class of Weak Keys for Blowfish, in: Fast Software Encryption, Springer, 2007, pp. 167–180.

[18] A. Trice, PhoneGap advice on dealing with Apple application rejections, [Online; accessed 2014-05-10] (2012).
URL `http://www.adobe.com/devnet/phonegap/articles/apple-application-rejections-and-phonegap-advice.html`