Maksim Riazanov

Development of a Cross-Platform Mobile Application for EIS

Bachelor's Thesis
Information Technology

November 2016

MAMK
University of Applied Sciences

DESCRIPTION

MAMK University of Applied Sciences	**Date of the bachelor's thesis**
Author(s) Maksim Riazanov	**Degree programme and option** Information Technology

Name of the bachelor's thesis

Development of a Cross-Platform Mobile Application for EIS

Abstract

This thesis follows the development process of the cross-platform application for NearMiss system. It describes application progression starting with the basic architecture. It explains development decisions in details and shows the development process as well as result itself. Specifically focusing different features implemented during the sprints it is possible to show why they were necessary, how they were implemented and what impact they had on the application. As this is hybrid application I have also tried to objectively compare upsides and downsides of such development approach. It is also in a way a demonstration of how different tools that make such approach possible are applied in practice.

Subject headings, (keywords)

Google Web Toolkit, Java, JavaScript, Apache Cordova, Hybrid, Cross-Platform

Pages 47	**Language** English	**URN**

Remarks, notes on appendices

Tutor Timo Mynttinen	**Bachelor's thesis assigned by** Observis Oy

CONTENTS

1 INTRODUCTION

In the modern world technology is everywhere and information is everything. There is an immense amount of data generated and processed all the time. It does not matter if it is entertainment, marketing, infrastructure, industry or science sector. There are only few spheres of human activity that will not benefit from the symbiosis of the classic approach and the integration of technology and the internet. As different industries have different goals and application cases for IT systems, it is quite hard to measure the general productivity boost of such an approach. However, it is natural that almost instant data transfer would improve human performance and communication in every sphere of activity, and there is a high number of additional advantages that are industry specific: greatly improved accountability, data security and integrity or overwhelming audience coverage, a multitude of management tools and automated data analysis. Such systems are called enterprise information systems (EIS) (Vandersluis,2013) and are used in different industries for a wide range of purposes. There are certain features that are important for every IT system in general, but especially when working in an enterprise environment these should be the focus:

- good scalability
- effortless integration
- data security and integrity

During the whole development course these goals must be kept in mind to ensure the proper planning and implementation. This and multiple other factors led to a decision of developing a mobile client for the NearMiss accident reporting system as a cross-platform, hybrid application. The main idea for this app is to be an addition for a server and a web-client with its own target audience. Only purpose is to make data input and transfer to the server simple and convenient for users. Big amount of planning and work was put in the user interface and experience, as the application is not only used by people who are not very comfortable with using such systems. Application also had to make a boring and tedious task easier, faster and more pleasant for users. During the development process the application also acquired several features that make it suitable for the wider range of industries.

2 DEVELOPMENT CONTEXT

The project I have been working with is called NearMiss. It is created and currently developed by Observis, Mikkeli-based company that often works on solutions that require the delivery of formatted relevant data to the end customer. Geolocation information is used to achieve improved relevancy of the supplied information. This time, however, it was the other way around. The goal of the project was to develop a system for a wide range of industries that would allow rapid and convenient delivery of accident or near miss accident (Accident that could have had a bad consequence, but luckily did not.) reports to the manager for further processing and storage of data. The latter part involves managers who receive and review the incoming reports so that they can address the reported issue and store reports for the further analysis and improved accountability. The system introduces tools that make the statistical analysis of stored reports easier by implementing different automated data mapping components available at the web interface. Users can send images, GPS data and their personal assessment of the situation in the report.

2.1 ADAPTIVE DEVELOPMENT

One of the main challenges of this project was the uncertainty of the final project requirements as the companies interested in this system had only a vague idea of what features and capabilities such software should possess. This was a huge restrain as our team had to work in sprints, implementing currently required features and functionality and to be very careful with our tool selection, as we needed to be prepared for the most unexpected feature requests and still have a good balance between ambiguity, performance and security. However, there was no need to drastically change our development course and technology stack because of feature requests. Still, when using the sprint approach, situations like that could have potentially happened. It is always a possibility, because some functionality can become critically important to implement in the product, and there is no other way around it. As an example, we did not use Ionic for our project because of the limitations of the UI components and because dynamic page generation could have been a requirement. Although it was still possible to make this feature work with Ionic, it would have had a negative impact on performance and a hit on usability. We also had no use for one of Ionic key features - different design for each platform that makes an application look closer to native. And, what is even more important, it had no official support for Windows Phone which was an explicit request from the beginning. Solution for this problem was using JavaScript + HTML + CSS as the standard of web development that could be heavily extended by libraries and Apache Cordova to make it available for different platforms as mobile application. Community has a huge number of open source libraries which are free to use and to achieve the desired results with as little overhead as possible. But, this can get quite messy in terms of code readability and debugging (e.g. libraries might have dependencies that conflict with each other). However, with this approach, developers can access a wider component selection and customize these components, if some specific behavior or look is required. At the same time if some library does not perform as desired, there is always an option to improve it or completely change it for something else.

2.2 TARGET AUDIENCE AND USAGE PRACTICE

Another big issue was that our application had to be used by a specific target audience. It is expected to consist mostly of people who are not comfortable and fluent when using their smartphones. Keeping that in mind it is safe to say that they are most certainly not confident using such applications. The conditions in which they are going to use this application are also far from perfect: e.g. in a factory during production, outside during cold and dark winter or in a place where no internet connection available. Such situations cause a lot of conditions that have a highly negative impact on user experience and application workflow. Therefore, there is an extreme need to anticipate such conditions during development and to design the application to lower the negative impact. This mostly results in UI changes, but some additional functionality had to be implemented as well.

3 TOOLS

To create the mobile application we used a unique workflow. Initially we create a raw and simple HTML layout. This layout is then injected with the actual complex HTML connected with the corresponding Java class. This is achieved by using the Observis proprietary library called xUI. This way we can manipulate DOM and add the desired logic directly in Java. This still only allows for basic logic and DOM manipulation as the library is quite recent and covers only essentials. To further amplify functionality and to add additional UI features it is possible to work directly with JavaScript. Libraries available for use in JavaScript are incredible and have a very different functionality to offer: from simple widgets to the full-blown language extensions and frameworks. It is also important to notice that Cordova plugins are JavaScript libraries in a way, so on this step it is possible to add hardware related functionality to the project (e.g. camera or GPS). The next step is Google Web Toolkit (GWT) compilation which translates Java into highly optimized JavaScript code. It later combines the existing JavaScript code with the GWT transpiled code, so in the end you have a big, unified JavaScript project. There are a lot of nuances which will be explained further in the corresponding chapters. After that we need to use Cordova, so that we can compile our project for the desired platforms.

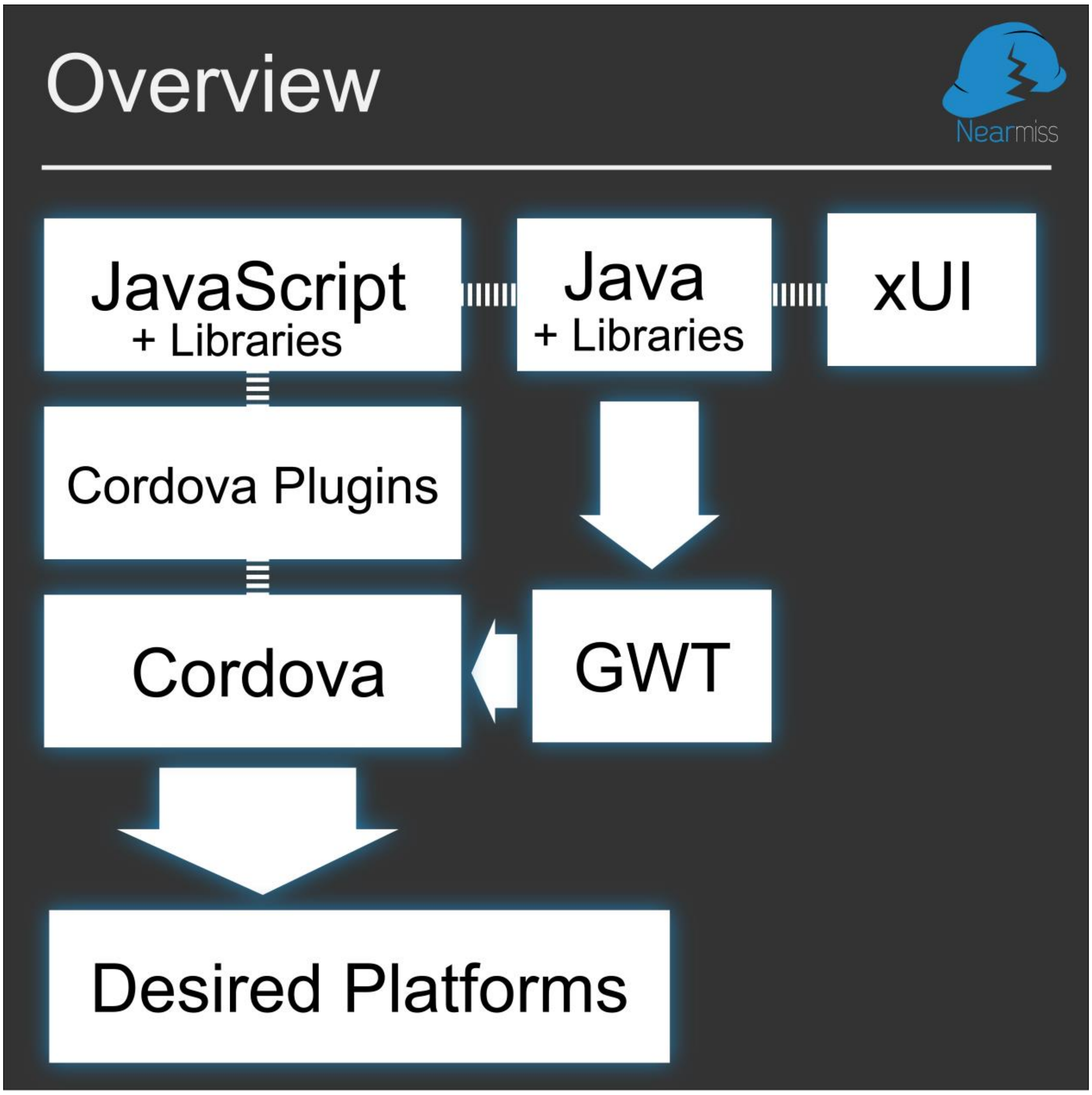


FIGURE 1. Workflow overview

Overall, this approach is not perfect as it requires a lot of preparation and precompilation as well as clear understanding of code logic flow as debugging is quite hard. On the other hand, it allows for great flexibility in component selection and implementation.

3.1 JAVA

Java is quite an old programming language. The Java version 1 was released in 1996 and has been growing ever since, introducing new features and improving older ones. Millions of programmers are using it, and the numbers are only increasing, as it has high demand on the market. Such popularity can be easily explained by the demand of Java applications in different industries. Java is used for enterprise development for the government or private structures, backend for the financial services, tools for software development, web applications, mobile applications and many more(Oracle, 2015). Such a vast number of appliances is the result of Java development team who made it possible to use the same code on different platforms regardless of their architecture (Oliver, 2013). This feature was quite an achievement at that period of time. This was possible with Java Virtual Machine (JVM) which is a part of Java Runtime Environment used to run compiled Java bytecode (Hoon Park, 2012). The slogan that describes such philosophy is "Write once, run anywhere" that was created by Sun. Official documentation describes JVM as follows:

"The JVM is an abstract computing machine, having an instruction set that uses memory. Virtual machines are often used to implement a programming language. The JVM is the cornerstone of the Java programming language. It is responsible for Java's cross-platform portability and the small size of its compiled code.
Solaris JVM is used to execute Java applications. The Java compiler, javac, outputs bytecodes and puts them into a .class file. The JVM then interprets these bytecodes, which can then be executed by any JVM implementation, thus providing Java's cross-platform portability."

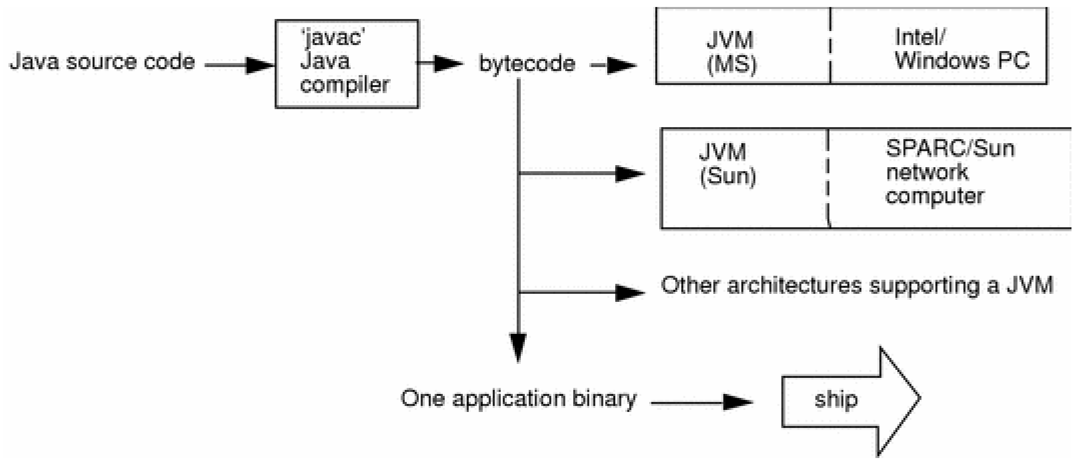


FIGURE 2. Java bytecode

This architecture approach made Java popular in early years because of its unique architecture independence. At the time being that was a huge issue as developers were required to program with a specific platform in mind.

3.2 HTML + CSS

Hyper Text Markup Language and Cascading Style Sheets have been together for a long time. They are responsible for the Internet, as we know it today (with the help of JavaScript). HTML is a markup language which basically means that it is used to describe the contents of web pages. To build a page developers need to use tags defined by specification, so the page will pass a validation process and can be viewed properly in all browsers. Though in the past, HTML and CSS combination was only used for basic markup and all the functionality was done in JavaScript. Throughout the time situation has drastically changed. Nowadays, the latest specification of the HTML has quite a lot of new tags that have functionality on their own, including but not limited to:

- <figure> element: makes it much easier to make a caption for image. It is now possible wrap an image and caption in that tag for automatic relative positioning (W3 Foundation 2010).
- contenteditable="true" attribute: makes element contents editable by users (Way, 2011).
- <header> and <footer> elements: headers and footers are no longer required to set by class, but became tags (W3 Foundation 2010).
- required="true" attribute for inputs: form validation will ask users to input proper value in such input before the form can be submitted (Way, 2011).
- <audio> and <video> elements: HTML5 allows native media playback and controls.
- embedded regular expressions: this feature allows to control user input in the forms directly from HTML (W3 Foundation 2010).
- data attribute: allows creating attributes with custom names and storing data in such attributes (Way, 2011).

Many features that required JavaScript in the past are now easily done in pure HTML.

CSS is used in addition to HTML to specify style properties of the document. It features the tag system with rules and logic that allows great flexibility when styling. It is also usually responsible for the proper positioning of elements on the page. As with HTML, CSS evolved over time, introducing more flexibility in tag logic and even more advanced features that allow not only to style but to generate HTML contents (Angelov, 2013). This next list shortly describes most significant new CSS features (W3Schools, 2015):

- Huge number of advanced selectors that we can use to create much more complex layouts with less code. These include selectors such as E:root - an E element, root of the document; E:nth-child(n) - an E element, the n-th child of its parent; E:first-of-type - an E element, the first sibling of its type.
- Function which allows arithmetic operations between most commonly used units in CSS. The calc() function can be used for dynamic calculation of media sizes and leads to better code segmentation and readability as it is possible to immediately see where certain dimension value comes from.
- Box Sizing rule which allows to changing, if padding and border are included in height. Though it may seem minor at the first glance, this addition helps to avoid a lot of problems when working in a complex layout.
- 2D/3D Transformations making it possible to animate contents directly in CSS. Transformation can be used for wide range of tasks: minor button feedback animation, positioning manipulation or complex animated scenes.
- Notoriously difficult column layout now available by adding simple CSS rule. New “column” rule makes it possible to easily create and customize column layout.

All these improvements make it possible to create low complexity sites purely in HTML + CSS and highly improve the development experience for complex projects. Such functionality is conventionally achieved using JavaScript. Quite a big drawback of the brand-new functionality and features is that they usually impair browser compatibility, it is very important to know and understand when and how it is possible to use them beneficially.

3.3 JAVASCRIPT

JavaScript is a programming language that is supported by all modern web browsers. Together with HTML and CSS they are responsible for the current look of the World Wide Web. Their combination is often called Dynamic HTML (DHTML), and it describes the usage of JavaScript as the main source of logic of web pages, being responsible for the background operations running on the page. It is also used to provide response for a user’s activities such as hover over hot zone that changes color, button visual feedback or a page change. The history of JavaScript is very interesting in the way that it is possible to observe how it evolved over time because of different circumstances.

The first release was created extremely quickly by Brendan Eich in just 10 days. The main idea behind this language was to complement and control Java applets(W3.org, 2012). The first release was in September 1995 with Netscape 2.0 under the name of LiveScript. The name was later changed to JavaScript as a marketing maneuver to make an affiliation that made the new scripting language look closely related to Java, as it was very popular at the time being. JavaScript quickly became independent, as people mainly used it not to work with applets but to bring interactivity to HTML pages (Champeon, 2001). Many professional programmers considered it a language for amateurs and not worthy of their attention because of its simplicity and lack of crucial tools: no compilation, no IDE or a reliable cross-platform debugging (W3.org, 2012). In that period, there were several attempts to implement a scripting language to introduce dynamic web, notably Microsoft was trying to reverse engineer JavaScript as JScript in Internet Explorer 3 (Champeon, 2001). It was impossible to create a robust solution which would work in every browser as each organization that used such scripting languages wanted to push their version through. Such competition was harmful for every company and standardization was necessary.

In later 1996th JavaScript was going through the standardization in the ECMA organization with the first release in June 1997. ECMA-262, officially called ECMAScript, became a standard, with JavaScript being the most widely used implementation with few less popular competitors such as ActionScript and JScript. After that, the development path was quite straight: ECMAScript feature set was growing with each version, number of different third-party implementations steadily rising, JavaScript still being the most popular and successful among them. After the release and adoption of Ajax techniques a lot more attention was drawn towards web development in general and JavaScript in particular. This attention led to development of high quality libraries and frameworks which further widened JavaScript functionality and areas of application. During the later development of ECMAScript there was some disagreement after ES4 was released. After the discussion, ES4 version was ditched and ES3.1 with some additions from ES4 became ES5. The proposal for the future development course and philosophy known as “Harmony” was released by Brendan Eich (Eich, 2011). The goals of that philosophy stated as follows:

1. Be a better language for writing:
    a. complex applications;
    b. libraries (possibly including DOM) shared by these applications;
    c. code generators targeting the new edition.

2. Switch to a testable specification, ideally a definitional interpreter hosted mostly in ES5.
3. Improve interoperation, adopting de facto standards where possible.
4. Keep versioning as simple and linear as possible.
5. Support a statically verifiable, object-capability secure subset.

JavaScript is currently taking one of the leading places in modern day programming with the wide application range. It is mostly so because of the number of different frameworks, libraries, techniques and development tools created over the time by the JavaScript developer community. Currently the most popular downloads on a JavaScript tools aggregator are React, Angular.js, Three.js, Ionic, Moment and Redux. This list includes not only small utility libraries (Moment) and the ones that introduce new functionality(Three.js,Redux), but also big frameworks that completely change workflow of JavaScript programming (React, Angular.js, Ionic). Because of such variety, it is extremely hard to distinguish single right development, compilation and deployment chain, as there are tools that alter these processes as well. It all depends on the desired goals and the approach: e.g. it is possible can create web application which will be compiled by browsers using their integrated compilers (IonMonkey for Mozilla, V8 for Chrome, Nitro for Safari). It is possible to later create a mobile or a desktop application, reusing almost all of the web application code but with few more steps. These steps include precompilation for the specific platform and an addition of code to support respective frameworks: Electron for desktop and Ionic for mobile.

3.4 GOOGLE WEB TOOLKIT

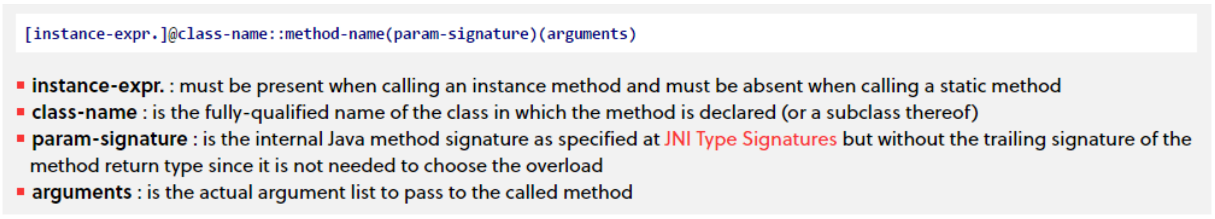
Google Web Toolkit (GWT) is an open-source toolkit that allows creating complex web applications in Java. It is used in multitude of web applications that require high performance and data-security as well as code reusability. To better understand how GWT is used in real world I would like to go thought the following example. The Development of Inbox by Google was a very resource demanding project and GWT helped to greatly reduce the development costs (Toubassi, 2014). The Inbox team had to develop for multiple platforms simultaneously and was expected to deliver the product that will outperform Gmail in every key aspect: UX (responsiveness, available features, stability), performance and personal data security. Therefore, they decided to develop the single data model and the application logic that will be used across all platforms. By using such an approach, it was possible to invest more resources in the critical codebase and to later have a more convenient and safe way to improve functionality. To achieve this, they used Java and GWT/J2ObjC for platform respective code transpilation. It was used to implement all desired functionality of application so that it later could be used to access all the required data and functions from the front-end.

Next, they had to transpile their base and add front-end for each platform. For Android, they did not have to make changes, as both the original base code and Android front-end is built in Java. For iOS, they used J2ObjC to transpile their Java code to Objective-C and to further create a native front-end for the iOS application. For the web application, they used GWT to transpile Java into JavaScript and to build an interface using more flexible and better looking components and solutions than currently available in GWT.

All in all, this is a great approach for a complex project with very high standards for quality as they were able to diversify their development process and at the same time share 60% of code across three platforms.
Although GWT allows achieving great results in plain Java, sometimes it is necessary to use a handwritten JavaScript. To do so, developers will need to work with JavaScript Native Interface (JSNI). It is extremely powerful GWT functionality which has a wide variety of application cases. It is possible to expose JavaScript methods for Java. This feature is great, because it allows to use self-written JavaScript code in Java project. It is also used to wrap the desired JavaScript libraries downloaded from external sources.

```
public static native void alert(String msg) /*-{
  $wnd.alert(msg);
}-*/;
```
CODE 1. JSNI wrapper example

JSNI supports not only a JavaScript to Java translation but a reversed one as well. In other word developers are able to access exposed variables and call the required code from the JavaScript. Because of the typing difference (Java has static typing and JavaScript has dynamic) exposing Java to JavaScript is somewhat limited and can be challenging, as the syntax is complex. The documentation has a generic example of such syntax (Google Web Toolkit, 2016).



```
[instance-expr.]@class-name::method-name(param-signature)(arguments)
```

- **instance-expr.** : must be present when calling an instance method and must be absent when calling a static method
- **class-name** : is the fully-qualified name of the class in which the method is declared (or a subclass thereof)
- **param-signature** : is the internal Java method signature as specified at JNI Type Signatures but without the trailing signature of the method return type since it is not needed to choose the overload
- **arguments** : is the actual argument list to pass to the called method

FIGURE 3. Exposing Java using JSNI

What is even more interesting, GWT transpiles Java code in a highly optimized JavaScript (Neethling,2008). Such optimization level is hard to achieve with hand written code. The optimization and compilation processes are complex and involve many steps to complete. This compilation process is a double edge sword, as it is the key feature of GWT and one of the main disadvantages of using it in development. The problem of this process is that it is quite slow, and therefore it leads to slower development cycle. Thus, it slows the debugging process and slows down development in long term by a noticeable margin.

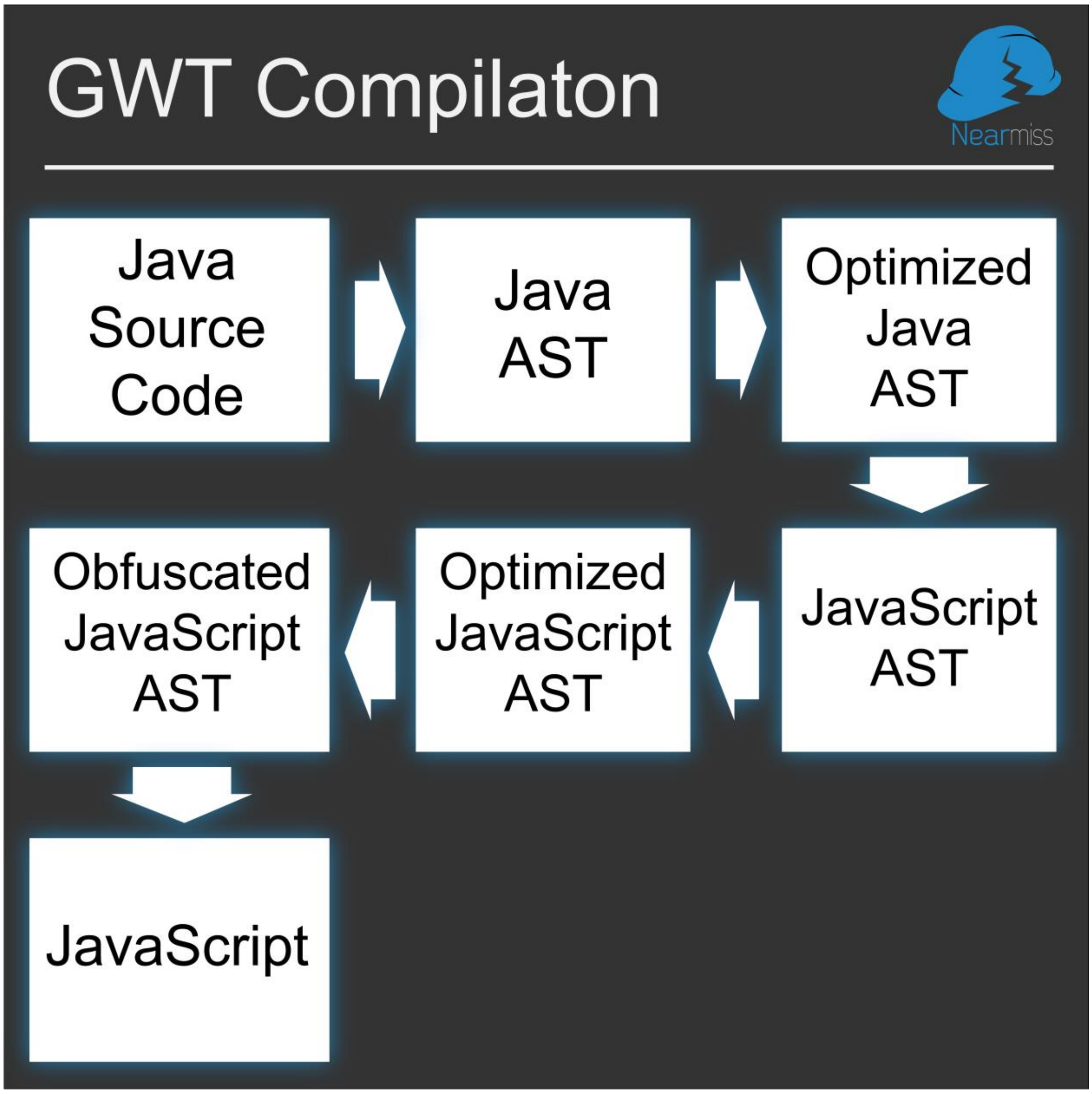


FIGURE 4. GWT Compilation (Buzdin, 2011)

AST is Abstract Syntax Tree and it is a representation of code that is after heavy optimization used to transform Java to JavaScript. There are multiple optimization techniques that are used in multiple passes, as new possibilities for optimization can appear during one of the passes (Buzdin, 2011). The Java AST optimization techniques include:

- Pruning removes unused and duplicate code and dependencies, drastically decreasing size of the output with no damage to functionality.
- Statification changes method to static if it does not depend on the instance methods and variables. This optimization allows to shrink RAM footprint of output code.
- Method call tightening changes polymorphic methods call process so that the exact required version of the method is called right away cutting the time required to distinguish the needed method.

- Method inlining moves short functions with no external references. Methods are moved to reside at the place of function call, as it is more efficient.
- Normalization transforms frequently used simple operations, so that they are optimal for further JavaScript transpilation. It also swaps JDK calls for the corresponding JavaScript transpiling optimized GWT library calls.

For the huge applications, there is also an additional opportunity to speed up the loading time by incorporating GWT code splitting (Google Web Toolkit, 2016). Code splitting is useful when there are pieces of the application that are not execution-time critical and are not used that frequently. If web application is modular, some modules have a very specific use case. It is possible to use this callback so that GWT will only load this part of the code for the client when it is required. Code is executed when asynchronous loading is finished.

```
GWT.runAsync(new RunAsyncCallback() {
          public void onFailure(Throwable caught) {
            Window.alert("Code download failed");
          }
});
```
CODE 2. Code splitting example

3.5 APACHE CORDOVA

This open source framework was formerly known as PhoneGap, developed by Adobe Systems. Later they have decided to release an open source version that is currently used in such projects as Telerik, Ionic, Intel XDK. Cordova is used to build a hybrid cross-platform mobile application. It enables developers to access hardware features with platform agnostic code. This means that one function for accessing, for example, vibration or brightness will work for all platforms. What is more interesting, Cordova features plugin system that makes it possible to use only required hardware features. Such modularity allows to have as little memory overhead as possible. It is also possible to create custom plugins that expose platform specific APIs to be used as desired or to implement required macros functionality (e.g. running some different native tasks consequently) (Cordova, 2016). As it is very demanding task to keep all the platforms APIs up-to-date, there are usually small quirks and intricacies that are caused by the difference in APIs and their continuous development process. These small bugs persist through many versions, but still Cordova allows building mobile applications in JavaScript, HTML5 and CSS with no need to study and practice languages for each separate platform.

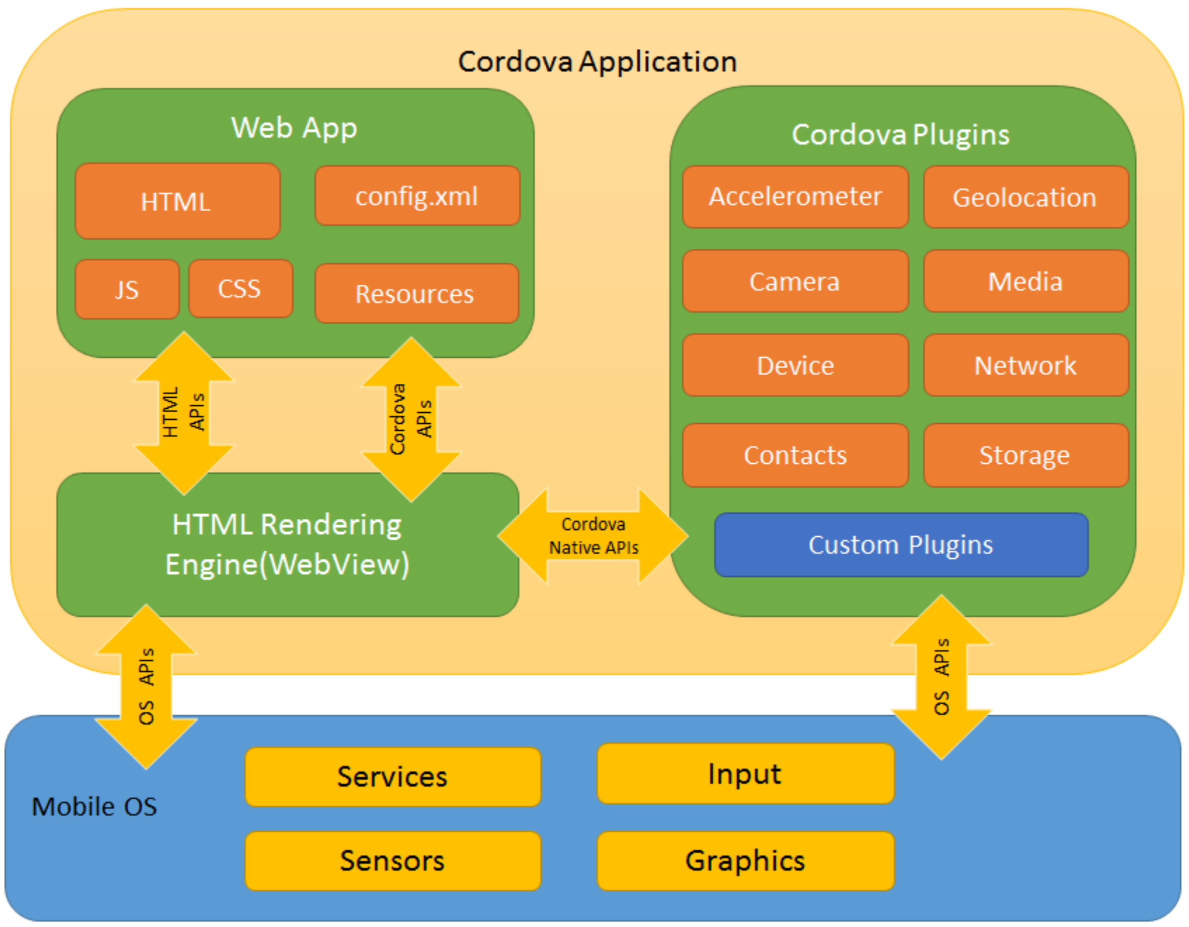


FIGURE 5. Cordova Architecture

It is important to point out that Cordova does not create a truly native mobile application, as it relies on WebView to render the layout. It does not transpiles HTML layouts to the native elements. This is the reason why creating a hybrid application is an extremely challenging task, as developers need to optimize their code as much as possible in order to achieve the desired performance (Reineke, 2015). Performance of hybrid applications has always been a notorious problem which has been slowly diminishing in the past few years with fast progression of web technologies. Modern web engines boosted up performance of the WebView based applications on mobile a lot. It is still very hard to develop in such context, as there is no consistency between different engines. It is always necessary to keep in mind things that work well for one platform might be completely unachievable for another.

4 IMPLEMENTATION

All parts of our team started working on NearMiss almost at the same time. Some base for the server and the web application was created in advance, but as I already mentioned we had no finalized feature requirements from the beginning. Therefore, all the parts of the project had to evolve simultaneously according to the required features. In such cases communication is crucial and right task priority and timely execution are the keys to smooth progression.

We have had some problems with communication in the beginning. One member of our team who worked on the server and web interface and who selected technology stack and prepared the base was working remotely. During the project kickoff time that team member was completely unavailable for about 1.5 months. It was a serious problem, because the server is the cornerstone for the whole project. The development speed of whole project was highly affected by this. We were getting behind our planned schedule, because REST API specifications that was left for us was not working as expected and we had to dig into the server ourselves to make the required changes. This was the biggest time fallback during the course of this project. Overall, it went smoothly as later we were able to synchronize and work as a team.

After some time, the person who was responsible for the server part has changed. After that our team was working much faster, as there was no more need for remote communication and we could react and adjust to new goals and conditions much faster. It was a very valuable experience as in such dynamically changing team and development environment it was possible to learn not only how to communicate and collaborate with colleagues that are available on the short notice, but also how to properly adjust to workflow, when some of your members are only available at certain periods of time.

4.1 PROTOTYPING

On the early stages of the project that revolves around user interaction it is crucial to prototype as much as possible. This allows producing number of different versions to see how ideas approximately look. It is important to further refine good solutions and use them in next prototypes to be sure that the idea works well. It is increasingly important when developing for the mobile, as screen space is very limited and the number of different resolutions is huge. Ideas that work for the smaller screen sizes may produce terrible results on even slightly larger screens and likewise. It gets even harder when there was no option to test something out quickly (animations, user interaction, performance of third-party component) and we needed to implement this feature to see if it fits in and meets our requirements. We were working through different versions very fast at the beginning, as it was mostly about figuring out what layout we should use for each application's screen. We could use mockups to quickly understand, if such layout has any good characteristics.

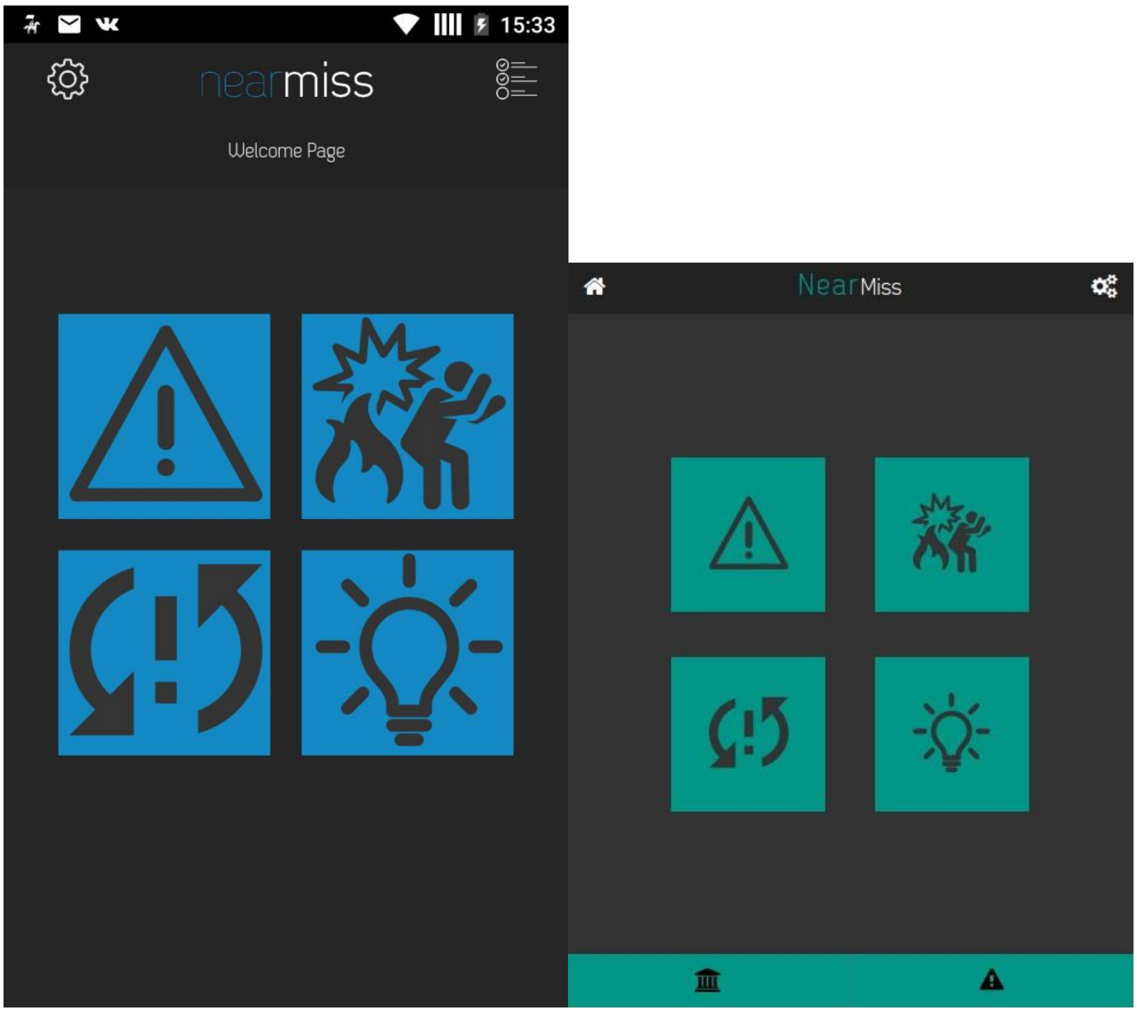


FIGURE 6. Main menu prototypes

When we started moving from the bigger and more general aspects into the details and small tweaking, it started getting harder and harder. Smaller elements in mobile development are very co-dependent and the number of things to consider was rising gradually: element sizing, precise positioning, color scheme, usability and uniform application look and feel. We also needed to figure out and keep in mind how any of these aspects may change in future and how to make these changes as fast and painless as possible. During the whole course of the project we never stopped the prototyping process as not only new features were always in development, but we were looking for a way to improve existing features by further adjustment.

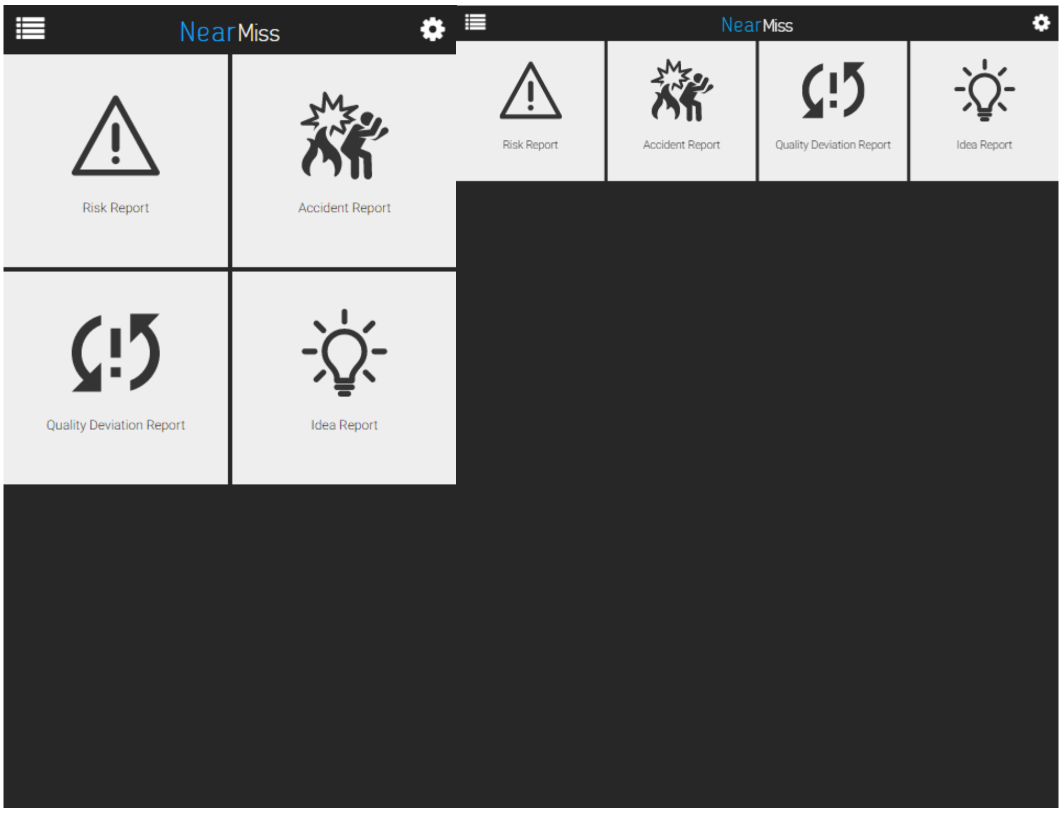


FIGURE 7. Final main menu

4.2 UI/UX

As our application focused on the simplification of user interaction with the NearMiss system, it was natural that a lot of planning and work was put in the development of the interface. The face of the application, what user sees, is critical for most application. Some developers prefer focusing on the application’s high functionality potential, and in some cases this might work, but it is highly situational. In my opinion, it is beneficial for every application and software to have UI/UX development as one of the main priorities. In the recent years, more and more attention given to the UI/UX development, as competition between similar software products increases. It is often so that the only thing that differentiates one software product from another is how pleasant it is for users to work with it. It gets even harder, as usually mobile applications are developed in addition to some existing base (web service, software, company etc.) that restricts the options even more. Base often involves branding of product, such as logo or fixed color palette. In big companies, there are designated teams that survey, research and process huge amounts of data to create techniques and solutions that are later used to improve different aspects of user interaction.

Good developer knows what users expect from their software, excellent developer knows what users wants before they do. Not being excellent developers or even good ones we still tried to do our best to ensure that potential users will not be reluctant to use our application. There were several different factors to consider but the critical points were the following:

1. Potential environment - Our mobile app is expected to be used in the industrial environments such as factories, warehouses or outdoors. There was no specifically targeted industry branch. It potentially could be used even a very remote location. Therefore, we always had to consider the worst-case scenario: ambient inconveniences such as bad visibility due to the low levels of light or view impairing particles such as snow or dust, non-uniform internet availability or special working gear that alters the normal usage scenario.

2. Target audience - It is expected that the main chunk of users will be workers of unspecified industry. This means that even though those people are familiar with using their smartphones, they are most likely not used to working with them for purposes other than basic phone functionality. It means that there is a great need to control the density of functionality per screen to avoid confusion or misuse and to maintain intuitivity of the interface. It is also very important to "explain" how application is supposed to be properly used by the provision of feedback and visual hints.

3. Usage scenarios - It was very important for us to understand that this application will be used in extreme conditions often involving high psychological strain, as it is tightly connected to accidents of all sorts. We wanted to ensure that our application’s look and user experience will not further irritate people that are already under pressure.

There were many ideas and solutions that might work for each of those cases independently but we wanted to make sure that all those problems were tackled simultaneously. We did not derived any new techniques, but applied already existing ones, so that it fits for our use.

Colors are extremely important, as they visible throughout the whole process of application use. Color palette selection is arguably the most complicated part of design. We wanted to make sure that we can make a thought through decision on that matter and use the color palette for our advantage:

Background color (#272727) - We decided to use a dark background, as it could address the previously described problems. It is not very commonly used in applications as a default setting these days, but for us it was a very attractive option. The application will probably be used mostly in Finland and the Nordic regions, meaning that light day is generally quite short. Darker background should be able to provide a better contrast and an improved visibility leading to less required brightness and battery consumption as a result. We were even considering going full black (#000000 hexadecimal color value) as it can reduce screen energy consumption for AMOLED screens (black color is achieved by turning off backlight of pixel in such screens). However, it looked too bad and did not fit our general look.

Primary color (#EEEEEE) - it is logical that in order to contrast with a darker color shades we needed to select a light color as the primary color for interactive elements of our application. We decided to stick with near white color, as it is neutral, attracts attention to the interactive elements and works well with every accent color.

Accent color (#0FA9F9) we used intense blue taken from logo that was presented as the initial project material. This color fitted quite well, being a good contrast to both background and primary color and fitting well to the whole composition and adding additional branding relation to the application.

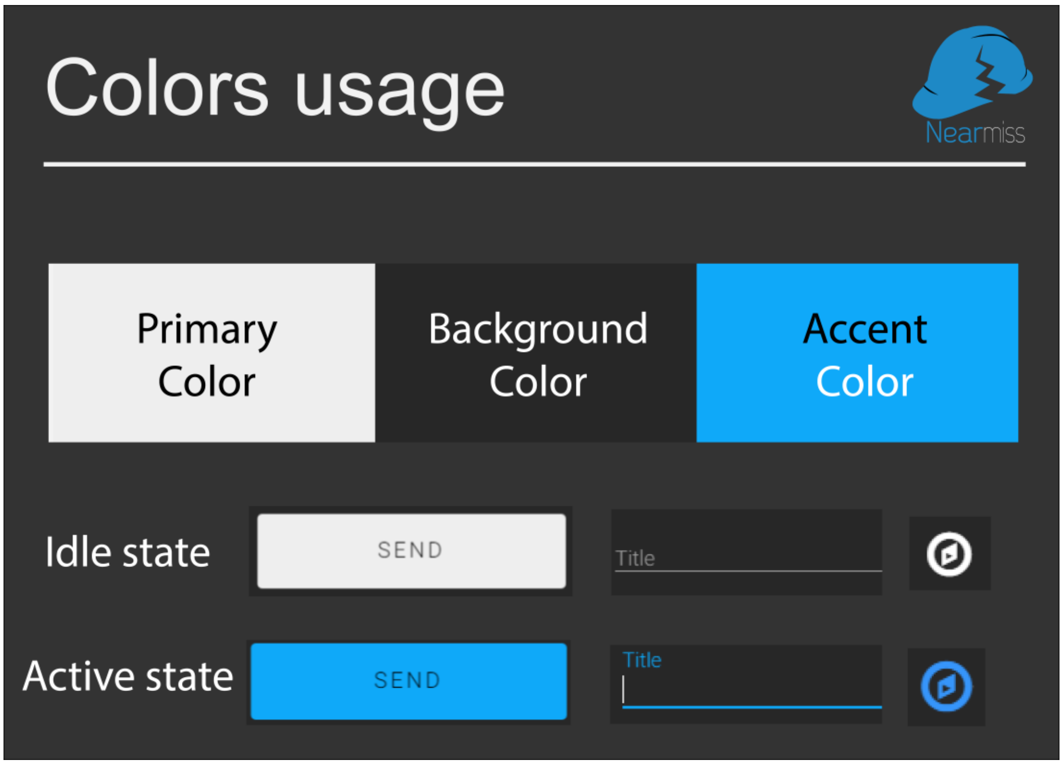


FIGURE 8. Color usage

4.3 FIRST ITERATION

As our team was working in short sprints, implementing functionality that was requested at the moment, we had a clear understanding of the task that we needed to do in the beginning of our project. The first iteration of the application only had to contain a raw, base functionality of the envisioned project. This included authentication mechanism, basic data input and communication with the server. It is a basic functionality that was not required to be extremely stable or feature rich at the time being but was greatly improved and extended throughout development process. To support involvement of the users we have created a list page, when user can observe their sent reports. When somebody working with application using manager's account it is possible to see all reports that belong to same company. This is done to improve situation control by managers as it is possible that they will not be behind their desk all the time.

4.3.1 AUTHENTICATION MECHANISM

The registration and login mechanisms are essential for every user centered application. Registration in the application was not required at time, because it was planned to be the task of remote managers to register users. This includes giving them out their credentials and introducing to system. This changed during the development process and will be discussed in "Registration" section. There are many factors that affect the selection of login method. One of the them is the decision if the third-party login systems that share credentials between a number of websites and applications should be incorporated. This is can be a highly effective technique in some cases, as the registration and login process is very tedious for users, and simplifying it is a great way to attract more people. Third party logins enable users to utilize their existing account from some other system. They can use their credentials to automatically register and login in the application. It is much more convenient for users to register and login in a couple of button presses and to connect with an already existing account. It might be somewhat dangerous, as there is no guarantee that the other service will not leak the credentials. There was a chance that the third party universal login system called Oath will be required later, so we decided to implement a home-made ambiguous login system.

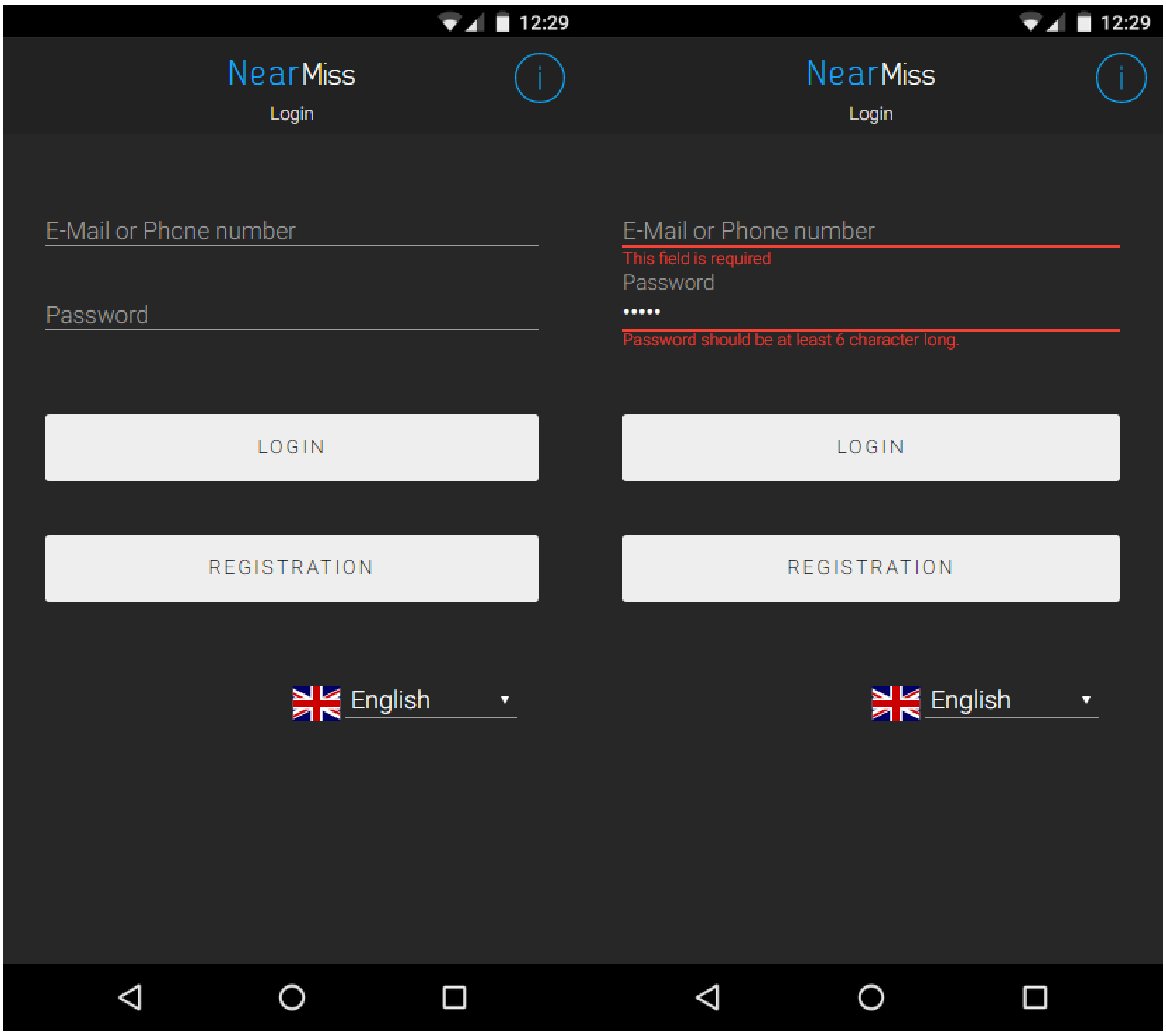


FIGURE 9. Login screen

We decided to use variable login options, meaning that users can login using their registered phone number or e-mail to authenticate. This flexibility is important, as we were not sure how managers want to proceed with user registration, and e-mail may not be used so often within our target audience, as it may be hard to remember if not used on regular basis. The password is required to be between 6-8 characters in order to make system less vulnerable to brute force attack. After the credentials are in the fields and the validation is passed (length and format) they are transferred to the server in JSON, using the jQuery.ajax() helper function. The server receives the information and compares it against the user database to check if the credentials are valid. If such a user is found and the password is correct, the server generates JWT (JSON web token) that is sent as a success response along with additional information that is later used by the application in further server communication. This web token defines what privileges a user has. The privilege level changes what information is available in the scope of user. Such approach required so that managers can have an instant access to all required information. If the credentials are valid, the user receives confirmation from the server and the

JWT token that are both stored for later seamless token renewal, as it has an expiration period.

```
function getToken(logindata, pass) {
   console.log(logindata);
   console.log(pass);
   $.ajax({
       type: "POST",
       dataType: 'json',
       url: dblink + "/mapi/mlogin",
       timeout: 5000,
       data: {
           username: logindata.toLowerCase(),
           password: pass
       },
       success: function(data) {
           if (data.user != undefined) {
               setData("email", data.user.email);
               setData("password", pass)
               setData("token", data.token);
               setData("firstname", data.user.firstname);
               setData("lastname", data.user.lastname);
               setData("company", data.user.company)
               setData("phone", data.user.phone);
               setData("pid", data.user.pid);
               setData("role", data.user.role);
               setData("dbid", data.user._id);
               setData("locations", data.locations);
               initLocationDropdown(data.locations);
           }
           callPopup("hide");
           changeTo("LandingPage");
           toast('Logged in successfully', 5000, 'toast-good')
           $("#pass-login").val("").change();
           $("#data-login").val("").change();



       },
       error: function(jqXHR, textStatus, errorThrown) {
           console.log(jqXHR);
           if (jqXHR.status == 0) {
               callPopup("loginCommErr");
           } else {
               if (textStatus = "401") {
                   $("#login-form").validate().showErrors({
                       "logindata": "",
                       "password": getConstant("generalLoginError")
                   });
                   callPopup("hide");

               } else {
                   callPopup("loginErr");

               }
           }
       }

   });
}
```

CODE 3. Login process

4.3.2 DATA INPUT

Main point of interest of application, is the page where user can input data. The First iteration created as a simple form only featured simple text and slider input but became feature reach during project course.

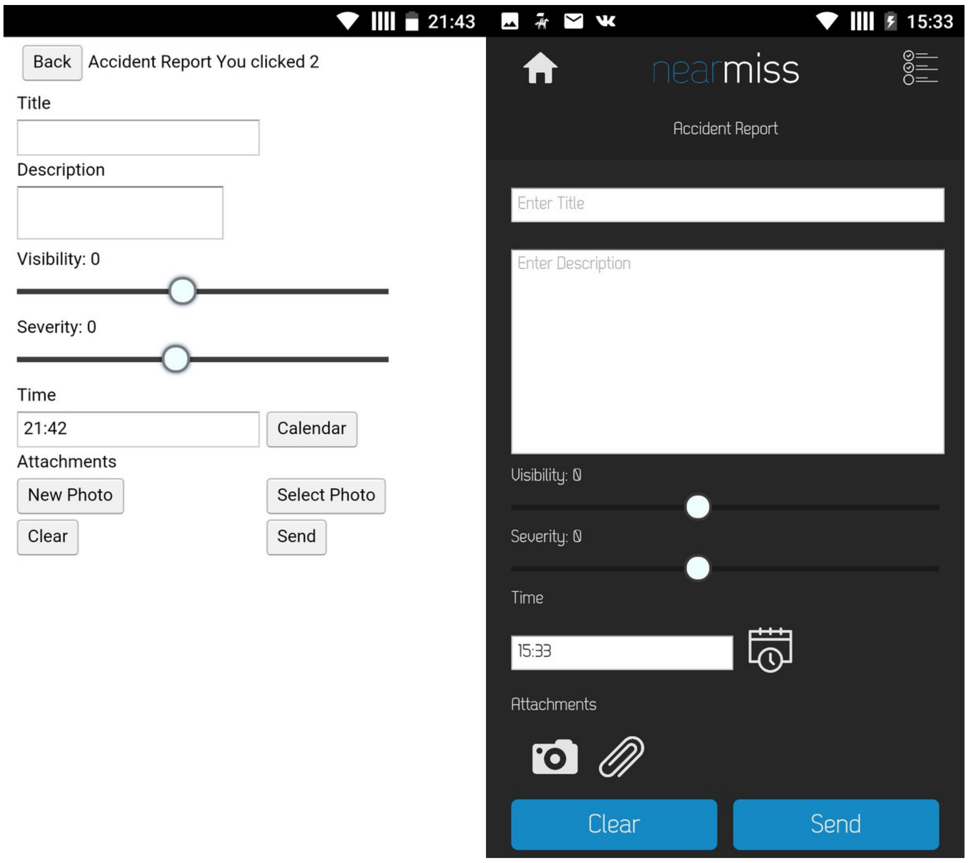


FIGURE 10. Data Input Prototypes

Still, the first version had to be responsive and feature a comfortable text input with further data collection. It was not hard to achieve, as there are numerous ways to get information from the single input elements or from the whole form at once. I used jQuery.serializeArray() to gather data from the text input elements in the form. I had to later assess and add values from the elements that can not be accessed by the serializeArray() function to get the full re-port. Data collection process got quite complex, when number elements and variability of scenarios increased.

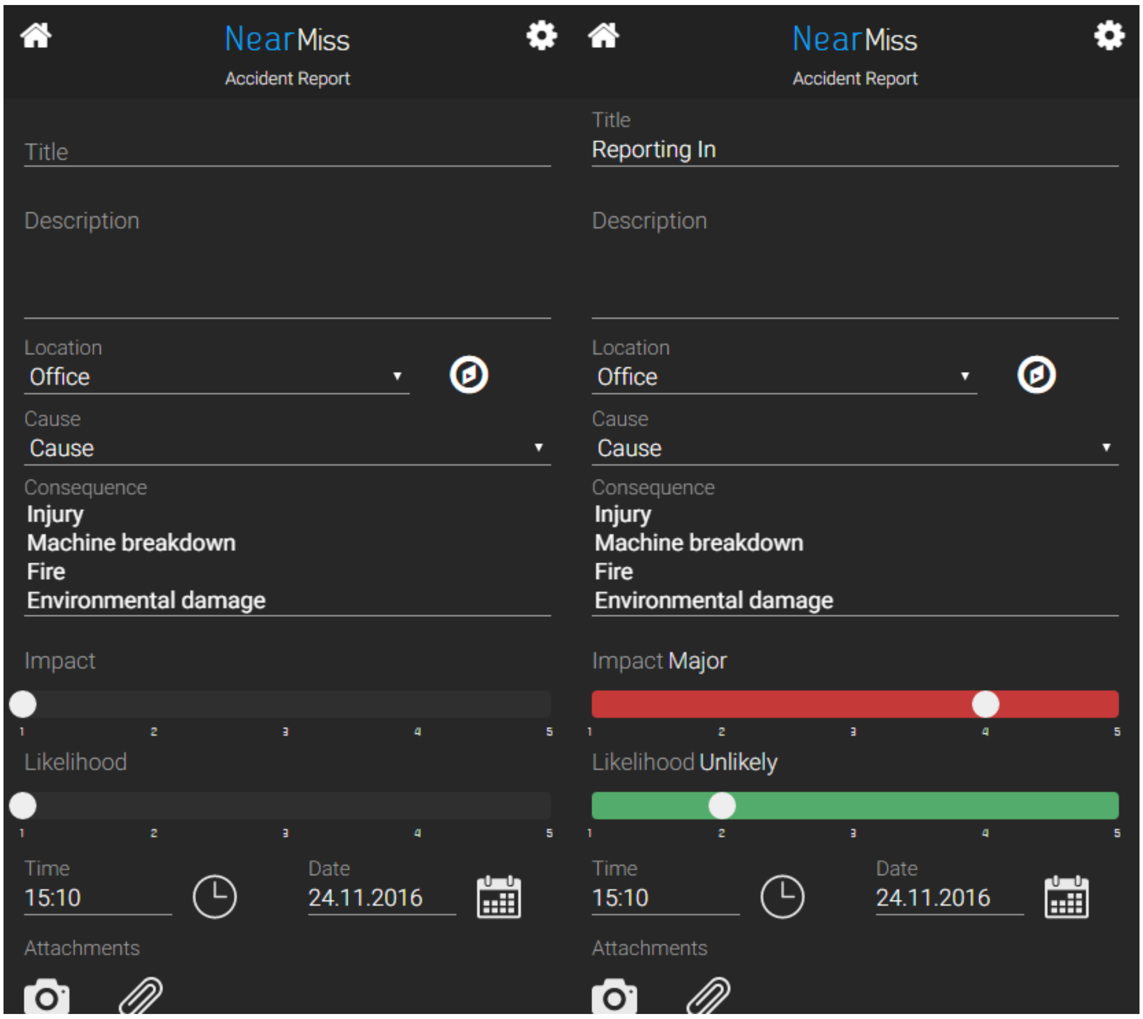
```
function buildJSON(formId) {
var json = {};
var formarr = $("#" + formId).serializeArray();

json.formtype = formId;
json.state = "sent";
if ($("#" + formId + "-slider-severity") != null && $("#" + formId + "-slider-
severity").length != 0) {
json.impact = $("#" + formId + "-slider-severity").slider("option","value");
}else{json.impact = "0";}

if ($("#" + formId + "-slider-likeness") != null && $("#" + formId + "-slider- like-
ness").length != 0) {
json.likelihood = $("#" + formId + "-slider-likeness").slider("option","value");
}else{
json.likelihood = "0";
}
return json;
}
```

CODE 4. Early report JSON creation

Later, even setting up initial look of forms was complicated as there was a number of elements that required initialization. These are elements that either add additional functionality such as time or date pickers, special sliders that work smoothly on mobile devices or elements that feature dynamic data that must be controlled and prefilled.



NearMiss
Accident Report

Title

Description

Location
Office

Cause
Cause

Consequence
Injury
Machine breakdown
Fire
Environmental damage

Impact
1 2 3 4 5

Likelihood
1 2 3 4 5

Time
15:10

Date
24.11.2016

Attachments

NearMiss
Accident Report

Title
Reporting In

Description

Location
Office

Cause
Cause

Consequence
Injury
Machine breakdown
Fire
Environmental damage

Impact Major
1 2 3 4 5

Likelihood Unlikely
1 2 3 4 5

Time
15:10

Date
24.11.2016

Attachments

FIGURE 11. Final data input

Initialization typically works by passing an element’s HTML selector that is used by library for binding, but as this is controlled by third-party library developers that is not always the case. In application, these initialization functions are used through JSNI as the lifecycle hooks of the application are much easier to access from GWT rather than from JavaScript. Initialization should be done on a step when, HTML already injected, but the application is still not visible to user.

```
private void JSInit() {
                          // Add stuff which has to be initialized with JS here
JSFunctions.createSliderJs("qualdev-report-form-slider-likeness","qualdev-report-form-text-
likeness",false);
JSFunctions.createClockJs("timeText");

JSFunctions.createCalendarJs("dateText");

JSFunctions.validateStartJS("acc-report-form");

JSFunctions.initLocationDropdownStorageJS();
JSFunctions.changeLangJS();

settingsPage.initFieldPopulationJS();
             }
```

CODE 5. Initialization process

At the later point of development process the data input was improved by adding date and time pickers for additional convenience.

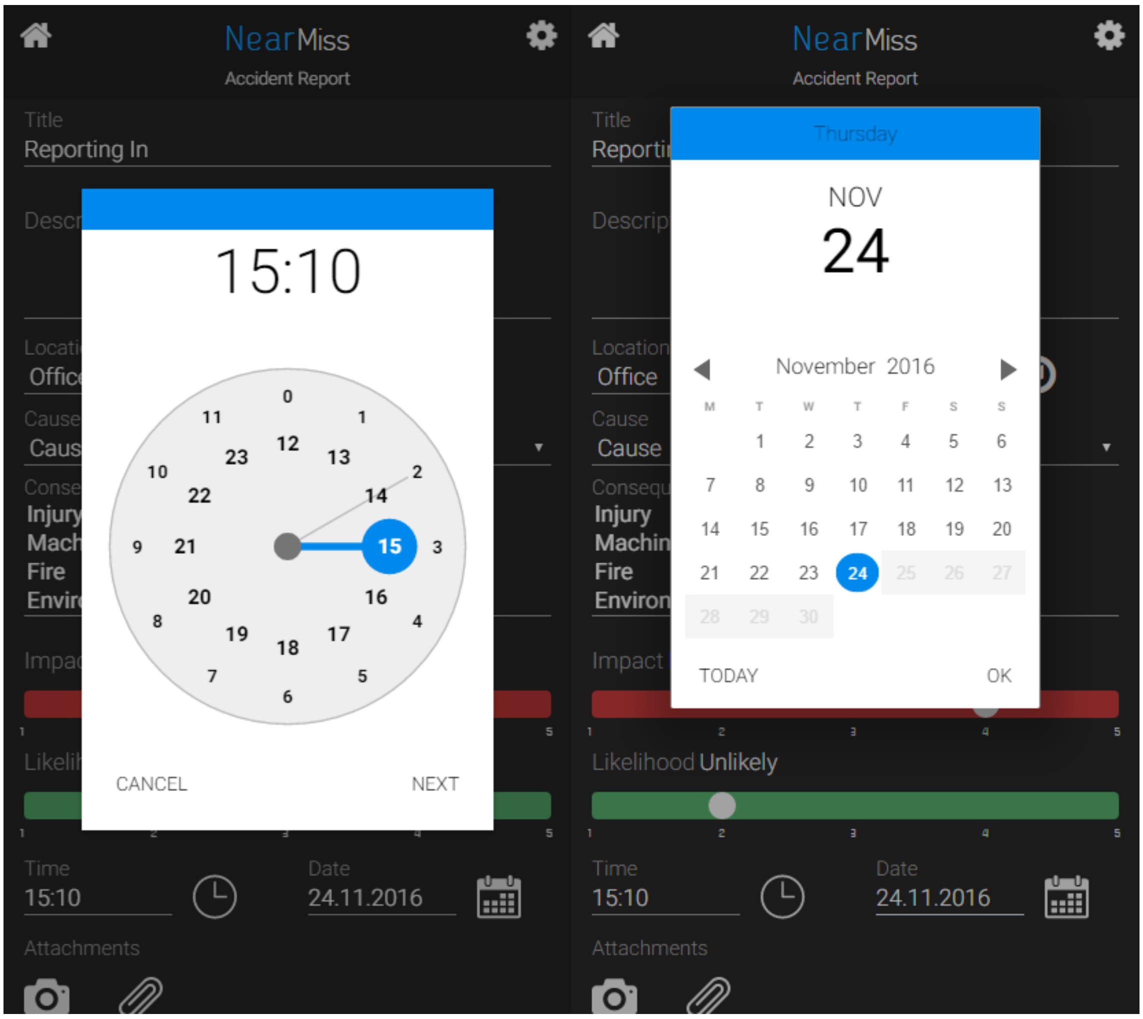
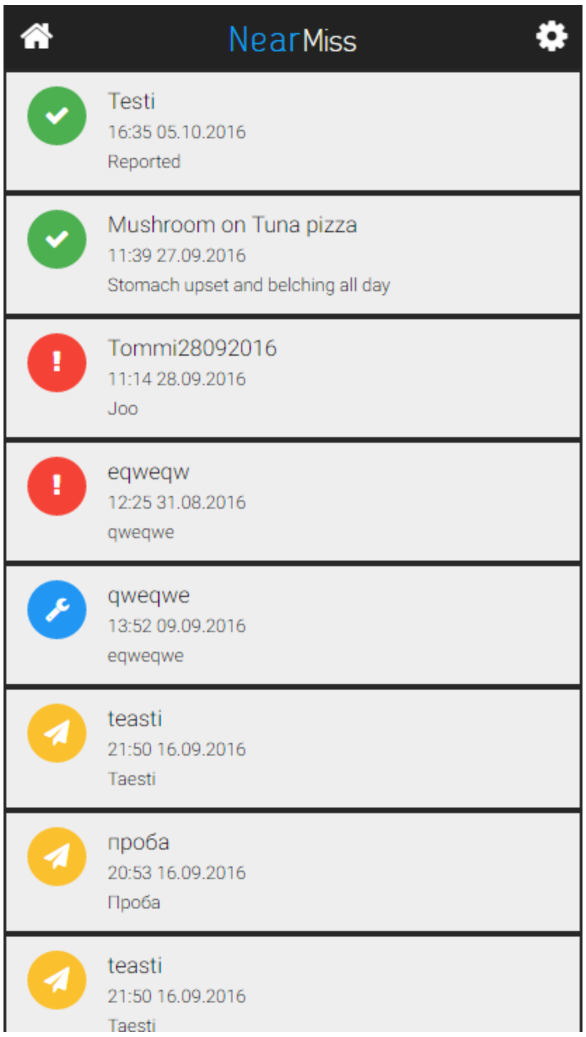


FIGURE 12. Time and date pickers

4.3.3 REPORT STATUS

In our application system, reports can have four different statuses. When user creates report and sends it to the server “SENT” status is assigned. When the manager views this report he or she can select the next state to assign. If it is possible for manager to immediately assess the situation and do something about it, managers can assign “DONE” state to the report. If there is more work to do, before some action is taken the report’s state is set to “IN WORK”. This state is also automatically assigned to the reports when the managers view them. If report state does not fit any of the category it is an exception and only intended for development process. State system like that makes it easier to control workflow among the managers. For users, it is much better experience to directly see what happens with their report. This increases the involvement in whole process and makes the whole process more interesting.



Testi
16:35 05.10.2016
Reported

Mushroom on Tuna pizza
11:39 27.09.2016
Stomach upset and belching all day

Tommi28092016
11:14 28.09.2016
Joo

eqweqw
12:25 31.08.2016
qweqwe

qweqwe
13:52 09.09.2016
eqweqwe

teasti
21:50 16.09.2016
Taesti

проба
20:53 16.09.2016
Проба

teasti
21:50 16.09.2016
Taesti

FIGURE 13. Report status page

In the status page it is possible to see shor information about report such as title, description and the timestamp. Icons on the left side represent the state of the report as following:

- Green – Report was reviewed by the manager and is now considered done. This means that manager executed all the necessary actions (such as calling some services or sending information further) and now finished with it.
- Blue – Report was seen by the manager and currently is in work. This means that the manager are working on addressing the issue stated in the report at the moment.
- Yellow – Report was sent and is successfully delivered to the server. This means that it now waits to be reviewed by the manager.
- Red – This state is an exception and is set by application when state in undentified. This is required for the development purposes.

Report list is obtained when users enters the page with the states. It is necessary to fetch the fresh data every time because situation may change quite fast.

```
function getReportList() {
        $("#list-loader").show();
        $("#report-list").html("");
        if (getData("token") != null)
                $.ajax({
                        type : 'GET',
                        dataType : 'json',
                        timeout:7000,
                        url : dblink + '/mapi/resources/reports',
                        data : {
                                access_token : getData("token")
                        },
                        success : function(data) {
                                buildReportListPage(data);
                                console.log(data);
                                console.log("Got Report list");
                        },
                        error : function(data) {
                                $("#report-list-
annotation").text(data.responseText);
                                console.log("Error Getting Report list");
                        }
                });

}
```

CODE 6. Fetch report list

After report list is fetch it must be rendered in the application. For that functionality basic HTML append is enough and there is no need to introduce templating libraries. It is also not so important to include template libraries because there is no need to escape the incoming text to avoid JavaScript injection as it is being filtered by the server.

```
function buildReportListPage(data){
        var list = $("#report-list");
        var annotation = $("#report-list-annotation");
        $("#list-loader").fadeOut();
        if(data.length == 0){
                annotation.text("You have no reports yet");
        }else{
                if(data.length == 1){
                        annotation.text("You have 1 report");
                }
                else{
                        annotation.text("You have " + data.length + " re-
ports");

                }
        $.each(data, function(i, el) {
                var $li = $("<li>", {
                        class : "collection-item avatar collection-item-
override"
                });

                switch (data[i].state) {
                case "new":
                        var $i = $("<i>", {
                                class : "circle yellow darken-2 fa fa-paper-
plane"
                        });
```

```
                                break;
                        case "in_work":
                                var $i = $("<i>", {
                                        class : "circle blue fa fa-wrench"
                                });
                                break;
                        case "done":
                                var $i = $("<i>", {
                                        class : "circle green fa fa-check"
                                });
                                break;
                        default:
                                var $i = $("<i>", {
                                        class : "circle red fa fa-exclamation"
                                });
                        }

                        var $span = $("<span>", {
                                class : "title"
                        });
                        var $p = $("<p>", {
                                class : "description-text truncate"
                        });

                        $p.html(el.reporttime + " " + el.reportdate + "<br>"
                                        + el.description);
                        $span.html(el.title);

                        $li.append($span);
                        $li.append($i);
                        $li.append($p);
                        $li.click(function() {
                                $(this).children(".description-text").toggleClass(
                                                "truncate");
                        });
                        list.append($li);

                });
                showStaggeredList(list)
                }
}
```

CODE 7. Rendering report list

4.4 SECOND ITERATION

After we had established the base for our application there was a need for ideas that support further growth and scalability of the application. During the first discussions, our team found out that it was planned to supply each customer with their own server that will exclusively facilitate customer's requirements. That was changed in the favor of one server that works for all customers at once. This could ensure easier scalability in the start of a system's life where the number of clients is controlled by Observis manually. So, there was a need to modify the data flow, so that all organizations will have a limited scope to ensure that data will not inter-leak between clients. In addition to that, we needed to create a new registration system as the manual registration of clients proved to be too tedious and time consuming.

The whole process of application usage was getting more complicated. This was bad as our target audience could have problems while using it. To make it more user-friendly, information page was created. It is available from the login screen. Information page contains basic instructions on how to use our application.

To facilitate a proper growth of application we needed to implement localization. As I was doing the application in English it was a very important task to implement the Finnish language to better suit our client’s needs, and we decided to make Russian localization just in case there will be an opportunity to work with Russian industries.

4.4.1 INFORMATION PAGE

Information page was required to provide users with additional hints on how to use application. It is targeting people who have very limited experience with such application. Information page available to access from the login screen. Such placement was selected so that users that are stuck during login or registration process will have some instructions to proceed.

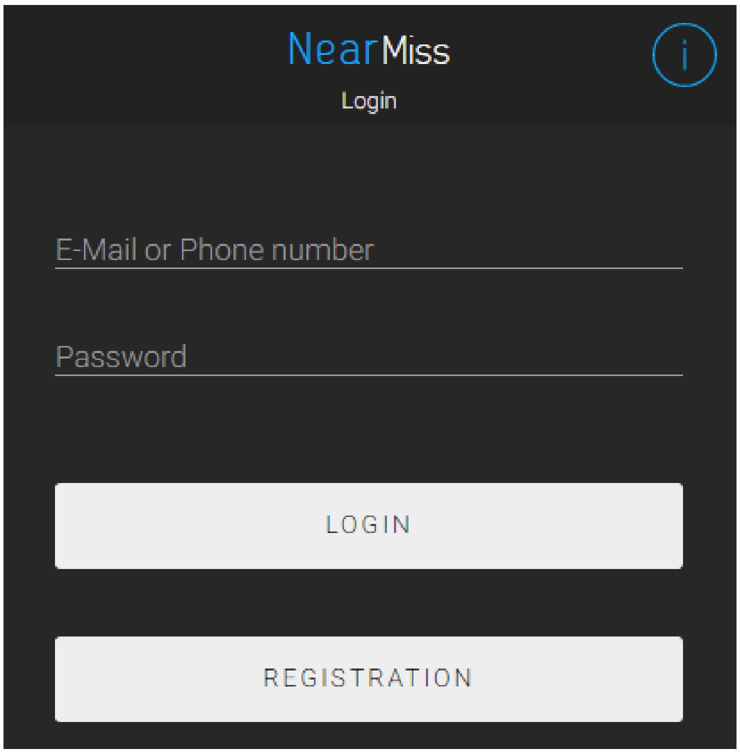
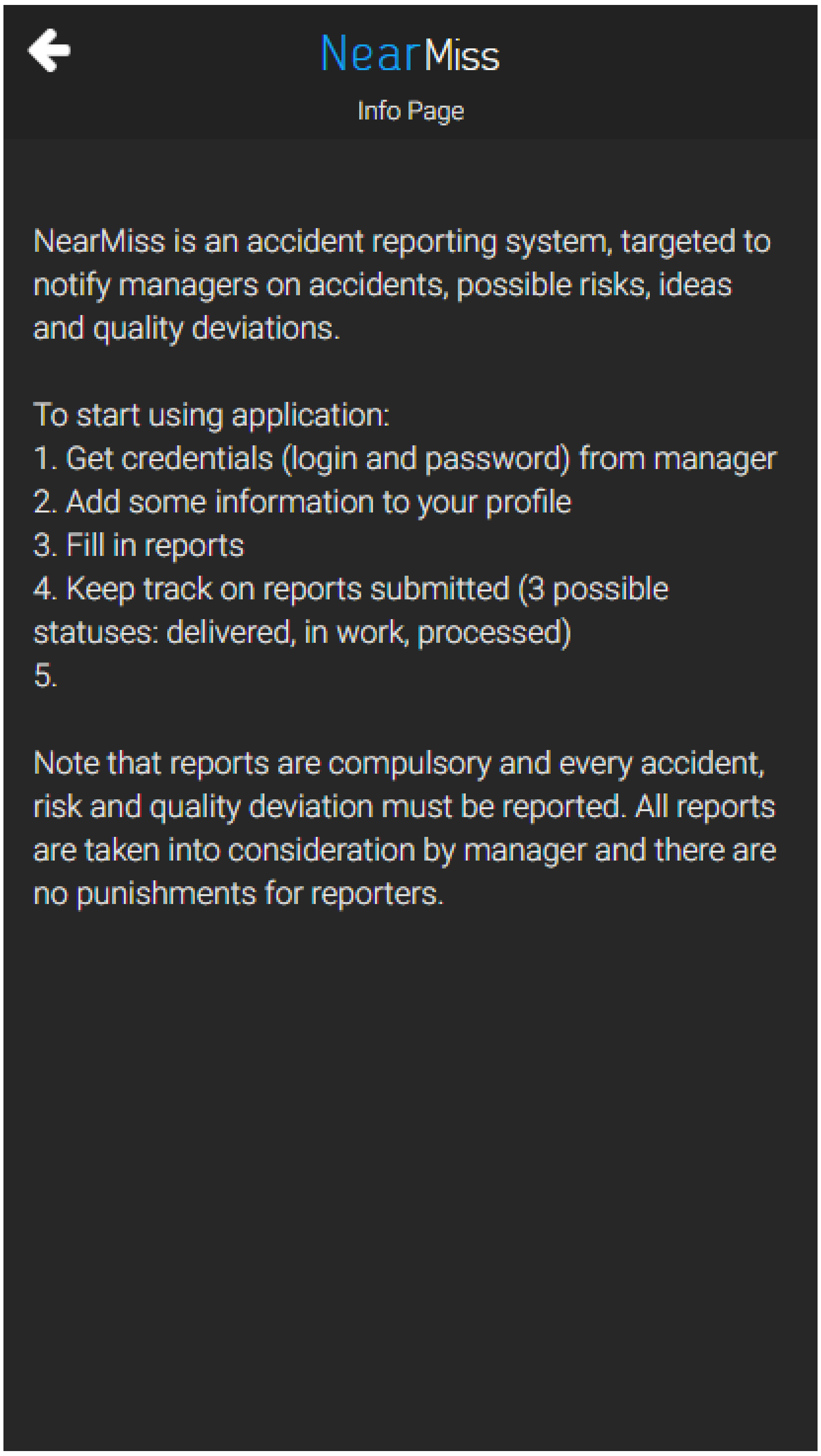


FIGURE 14. Info button placement

Perfectly, tutorials like this must consist of slideshow introduction when opening application. Compared to text explanation, slideshow illustrates the whole process of application use. Our team had other priorities at the time being and we have settled for the text instructions.



NearMiss is an accident reporting system, targeted to notify managers on accidents, possible risks, ideas and quality deviations.

To start using application:
1. Get credentials (login and password) from manager
2. Add some information to your profile
3. Fill in reports
4. Keep track on reports submitted (3 possible statuses: delivered, in work, processed)
5.

Note that reports are compulsory and every accident, risk and quality deviation must be reported. All reports are taken into consideration by manager and there are no punishments for reporters.

FIGURE 15. Info page

4.4.2 DATABASE REFACTORING

As I mentioned earlier there was a requirement to change the data flow on our server and the communication with the application so the client companies' data reside on one server. It was important, because we needed to strictly limit the visibility scope of each organization residing on a server. As we were working with MongoDB, the most obvious solution would be dividing the database at the root by creating a different subdirectory for each company. All users and company related information reside in this node. Instead, we decided to stick with a more flexible solution and created, an additional directory with companies alongside the main user pool. Every user was assigned an identifier to be associated with a certain company. This means that there is a pool of all users that are later sorted during the REST API request process when necessary. This approach allows for high variability of user filtering and does not restrict exceptions such as users with multiple companies or a parent company that contains multiple companies inside it. It also makes easier to implement different filters for statistics.

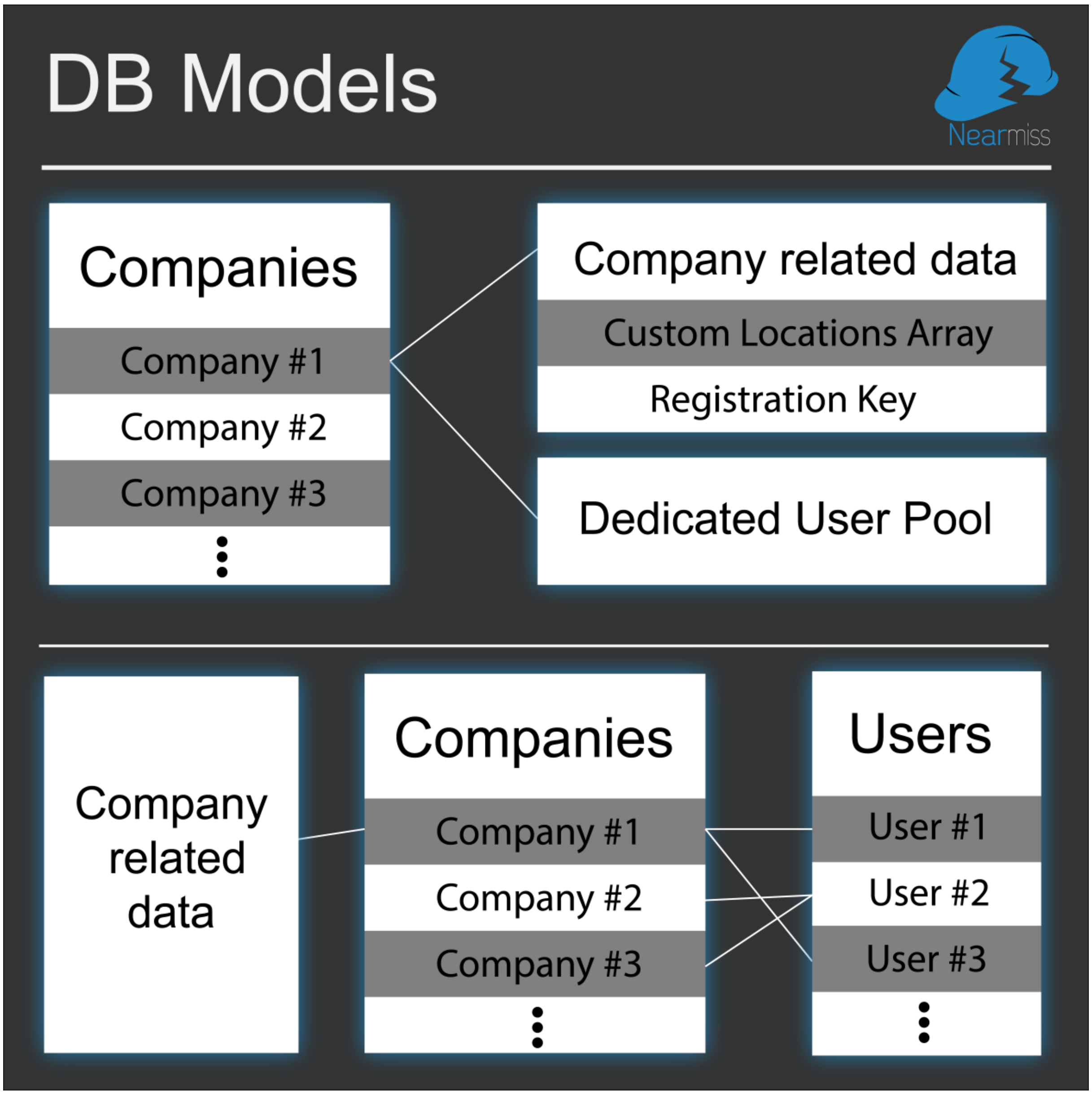
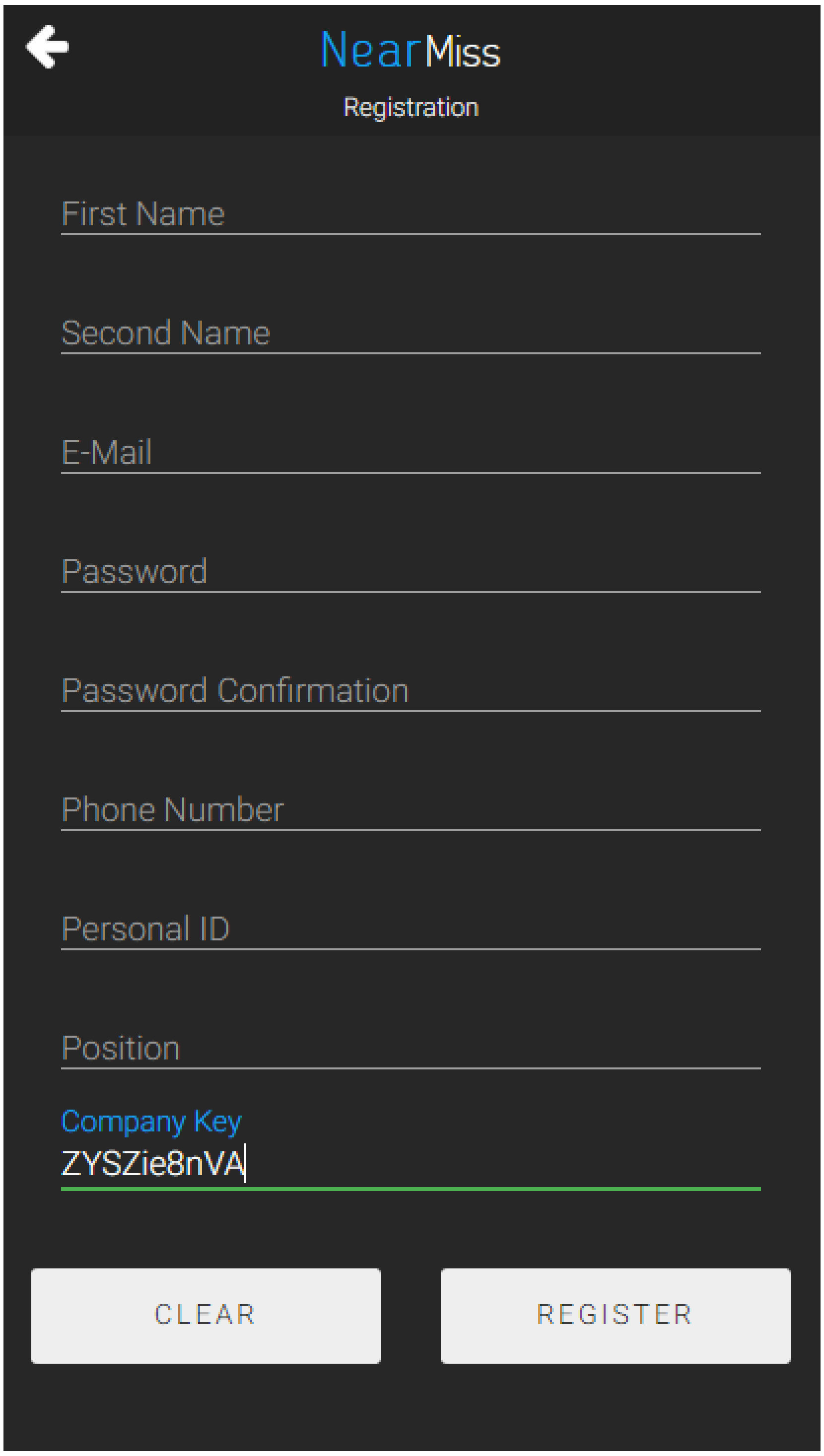


FIGURE 16. Database models

4.4.3 REGISTRATION

To simplify registration our team implemented the system that decentralizes the process from the manager and moves responsibility to users, but still allows for the regulated and controlled registration. The idea is that user registration for systems like NearMiss is usually carried out at certain periods of time. Therefore, there is no need for it to be always available. So, when everyone that needs to use the system is registered, registration becomes unavailable, so that there won't be any additional users. This helps to avoid errors that may result in data leaks or malicious exploitation. We achieved this by allowing managers to freely open and close registration, when there is a need to do so. When registration is open, the key is generated and accessible at the web application by a manager. Later, this key is given out to users that must access the system. They can use this key to register within the application and enter all the required data that otherwise would have been the responsibility of a manager.



Company Key
ZYSZie8nVA

FIGURE 17. Registration Page

When they are using the generated key, system automatically associates a user with the company which has given out their respective key. With such a system, it is possible to have semi-automatic controlled registration, so that the managers will only need to monitor the whole process and make sure there will be no unknown registered users. This simplifies a managers' work a lot, as there will be no need to manually register each person.

4.4.4 LOCALIZATION

Localization is the process of adaptation for the target market different from the original. There are different approaches for localizations, as it all depends on what market is targeted, how the difference between original and targeted market affects the product and approach of the parent company. The most basic localization approach involves translation onto the native language of the targeted territory to ensure usability and attractiveness of a product for broad audience. It might prove quite difficult to adapt the product for the market that has some major differences compared to the originally targeted market. Such difficulties are often caused by crucial cultural differences resulting in completely different mindsets and priorities of potential clients. Hopefully, that was not a problem for our team as we were targeting Finland and neighboring countries.

```
private void initLanguage(){
                String language = JSFunctions.getDataJs("language");
                if (JSFunctions.getDataJs("language") == null) {

                        JSFunctions.setDataJs("language", "fi");
                        language = "fi";
                        Re-
sources.getConstants().setCurrentLanguage(language);

                } else {

                        Re-
sources.getConstants().setCurrentLanguage(language);
                }
        }
```
CODE 8. Language initialisation

Since there are no extreme differences translation only is enough for localization. We made a Finnish language default and left English and Russian as additional options. Additional options very important, as there are a lot of international workers present. Localizations in applications are usually implemented in such way that instead of using plain text, developers need use the function which should return required text that has a variable source data. In our application, this is handled in Java by xUI library. Using xUI is possible to create text constants that relate automatically relate to the functions by naming. It is possible to further use that function to get associated text value in your Java Code or use JSNI to use it in JavaScript.

Although there are plenty of pure JavaScript localization solutions available, it is still much more convenient to wrap the Java code for reuse. It is mostly used when creating layout in Java with few JavaScript exceptions.

```
public void onInit() {
                riskReportText.text(constants.riskReportHeader());
                accReportText.text(constants.accReportHeader());
                qualReportText.text(constants.qualDevReportHeader());
                ideaReportText.text(constants.ideaReportHeader());
}
```

CODE 9. Main menu text initialization

4.5 THIRD ITERATION

During the third sprint our team focused on improving the report creation process. We have introduced new features and improved already existing ones to better facilitate clients' needs. Changes were made to introduce the logic of sending attachments and the report offline storage. Our team also worked on the major overhaul of location field in the report. This overhaul changes the location selection field usability, greatly improving the functionality of the latter and incorporates GPS custom location. These changes allow for a much flexible use of the application and make it less dependant on the use scenario. Such flexibility is important to attract new industries that could not efficiently use the system otherwise.

4.5.1 LOCATION

To start with I would like to explain how location is used in our system. When an incident is reported, users should select some location to mark where the incident took place. In facilities that have many production sites and sub-areas it is important to properly associate incidents with the specific isolated areas. First, the manager is able to better understand the currently ongoing situation context, and therefore to react faster and with more precision. Secondly, this information can later be used to recognize the most dangerous areas by the association with accident impact level. This information can be used to issue counter measures accordingly, leading to correct resource distribution and improved accident prevention.

Initially our team started working on a very basic custom location field as a request from one of the potential clients, but as soon as we realised how much potential this feature brings, we expanded it greatly. Using the web application, managers can create custom locations that will be available for every user in the same company. Users can select one of the locations when creating the report to specify where incident took place. It is more convenient for users to select the location from a predefined list than to manually input a location name. It is also less effective, as there would be naming difference or errors making it harder for managers to control the situation.

What happens in practice is that when the manager adds or deletes a field on the web interface, an array of locations belonging to a company is being modified. Later, whena user logs in, the application receives this array, stores it for further use and sets up location selector. Locations are renewed with the JWT token, as there is no need for real-time update, as these locations will not change often.

```
function saveGeo(boxId) {
        var box = $("#" + boxId);

        if (box.hasClass("glow-anim")) {
                return false;
        }
        if (box.hasClass("locked")){
                box.removeClass("locked");
                box.removeData("geodata");
                return false;
        }

        box.addClass("glow-anim");

        navigator.geolocation.getCurrentPosition(function(position) {
                        box.removeClass("glow-anim");
                        box.addClass("locked")
                        box.data("geodata", position.coords);
        },
                        function(error) {
                                box.removeClass("glow-anim");

                                callPopup("Location unavailable\nPlease,check
location settings of your device")

                        },
                        { maximumAge: 3000, timeout: 15000, enableHighAccu-
racy: true });
}
```

CODE 10. Getting and storing geolocation

To further improve this system we have implemented custom, user-provided GPS locations. For some clients such a system is crucial, as the working environment is dynamic, and there is no possibility to provide a list static locations. It can also be used in addition to the regular location selection to provide additional information.

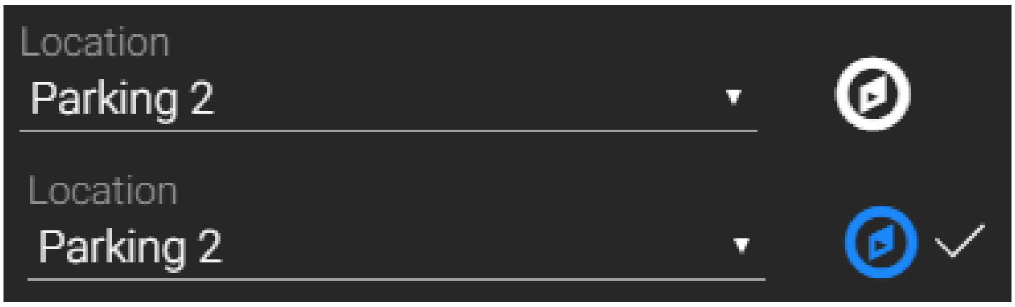


FIGURE 18. GPS location button

We have used a Cordova Geolocation plugin to gather information. It allows accessing hardware API while maintaining cross-platform code sharing. The geolocation function returns a set of values that can be used to identify a location and its accuracy. The end result is dependant on the hardware, as older devices might have lower accuracy. General success callback of the Cordova Geolocation plugin ”getCurrentPosition” function contains: latitude, longitude, altitude, accuracy, altitude accuracy, heading, speed and timestamp. In our case we only required the latitude, longitude and accuracy values to get necessary positioning information.

4.5.2 IMAGE ATTACHMENTS

Capturing and sending images is an important part of the reporting process. Sending an image is sometimes much more convenient than trying to describe the situation in writing. It is safe to assume that every modern smartphone has a decent camera that can be utilized to capture images. It is also possible to attach images taken in advance to the report. The Cordova Camera plugin allows accessing the device’s hardware to make it possible to share the same code base across different devices.

```
  navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
    destinationType: Camera.DestinationType.FILE_URI });

function onSuccess(imageURI) {
    var image = document.getElementById('myImage');
    image.src = imageURI;
}

function onFail(message) {
    alert('Failed because: ' + message);
}
```
CODE 11. Example getPicture()

There are lots of different options available for developers to control the snapping process. They are set in the ”CameraOptions” object and passed as the parameter to the function that activates the camera. These options allow controlling various resulting image properties and the process of taking the photo itself:

- Quality – Is the number expressing quality requirements for image. It varies from 0 to 100 and represent quality percentage. This means that 100 is a full resolution image with no compression applied.
- Destination Type – This option sets the return format for the ”getPicture” function. It can either be DATA_URL, FILE_URI or NATIVE_URI. The DATA_URL request returns the image encoded as base64 string. This can be used in some cases, but generally it is not recommended, as it is very resource demanding operation. The FILE_URI and the NATIVE_URI options return a link to the image in different formats.
- Source Type – This option defines the process of acquiring an image. It is possible to select from PHOTOLIBRARY, CAMERA and SAVEDPHOTOLIBRARY. The CAMERA option is used when there is need to take a new image and obtain it. The PHOTOLIBRARY and the SAVEDPHOTOLIBRARY options only different for some platforms. On some platform the SAVEDPHOTOLIBRARY options opens up only recent camera roll and not all the available images on device.
- Allow Edit – This option can be true or false. It toggles user’s ability to edit the image before it is returned to the callback function.
- Encoding Type – This option allows develeopers to select between JPEG and PNG return image formats.
- Target Width and Height – These values make it possible to specify resolution of image that will be returned in the success callback. This will not affect the image resolution saved in the phone’s internal memory.
- Media Type – the Cordova Camera plugin allows not only taking pictures but to record videos as well. This option makes it possible to specify explicitly what is expected in callback function. PICTURE, VIDEO and ALLMEDIA options are available.
- Correct Orientation – This option can be true or false. When this option is enabled, it fixes issues with image processing for some platforms. Sometime it may occur that when horizontal image is snapped it returned as vertical but with wrong orientation.

- Save to Photo Album – This option can be true or false and it specifies if image should be saved or discarded after it was returned to success callback.
- Popover Options – This is an iOS specific option that controls the position of the image selection menu on iPad. The options available are x and y coordinates of the anchor element for popover, width, height and arrow direction.
- Camera Direction – This specifies either frontal or rear camera should be activated by default when taking a picture. The options available are BACK and FRONT.

It is possible to see that this plugin is extensively covering the process of accessing media. Such option variety makes it easier for developers to customize user experience, as they see necessary.

In the NearMiss application I used a returned image not as an end result but just to display it on the thumbnail element. It is done so that users may decide to delete some images and so that all the report data is grouped nicely in one page.

```
function displayThumb(imgUri, elemId) {
        var box = $("<div/> ", {
                "class" : "photo-thumb",
                back : "Click me!",
                click : function() {
                        $(this).remove();
                }
        }).appendTo('#' + elemId).css("background-image", "url('" + imgUri
+ "')");

        box.data("srcURL",imgUri);
        var cross = $("<span/> ", {
                "class" : "icon icon-cancel"
        }).appendTo(box);

        cross.css({
                top : box.height() / 2 - cross.height() / 2,
                left : box.width() / 2 - cross.width() / 2
        });

}
```
CODE 12. Thumbnail creation

Afterwards I use link from these thumbnail elements to get the image from device memory and to send it to the server. Images are sent after the report is successfully accepted on the server. It responds with callback, so that it is ready to receive and associate images with a report.

4.5.3 OFFLINE STORAGE

Offline functionality was extremely important to support the desired level of usability for our application. Storing data itself is not hard when using Cordova, as it allows using the same storage methods as any browser. What was seriously concerning is that using simple local storage allows to only store small amounts of data, and we did not know how to proceed with storing images. Hopefully we found out that it is possible to make Camera plugin save images in the phone’s memory and to store only a link that is later used to access image. However, we still needed to incorporate a system to detect the ”Network offline” state and properly handle sending reports while offline. We decided to use the error callback on report sending AJAX request to detect connectivity issues. Cordova offers a callback on when network status changes, but there are no means to check, if the connection is present at the moment. It is only possible to achieve by pinging some designated server and it is horrible idea to do. So that is why it is currently works so that if the report dispatch failed, it is stored offline.When Cordova fires ”Network online” callback application automatically uploads all pending reports to the server in the background.

```
function sendFromStorage(reportArr) {
    if (reportArr == undefined || reportArr.length == 0) {
        return;
    }
    json = reportArr[0];
    json.access_token = getData("token");

    if (json.attachments != undefined) {
        var attachments = json.attachments;
    }
    delete json.attachments;
    $.ajax({
        type: "POST",
        dataType: 'json',
        url: dblink + "/mapi/resources/report",
        timeout: 2000,
        data: json,
        tryCount: 0,
        retryLimit: 3,
        success: function(data) {
            // show content
            reportArr.shift();
            setData("reportCell", JSON.stringify(reportArr));

            if (attachments != undefined && attachments.length > 0) {
                sendAttach(attachments, data.reportId, null, null);
                sendFromStorage(reportArr);
            } else {
                sendFromStorage(reportArr);
            }

            console.log(data);
            console.log('Send from Storage Success!');
        },
        error: function(jqXHR, textStatus, errorThrown) {
            if (textStatus == 'timeout') {
                this.tryCount++;
                if (this.tryCount <= this.retryLimit) {
                    //try again
                    $.ajax(this);
                    return;
                }
                return;
            }
            if (jqXHR.status == 500) {
                //handle error
            } else {
                //handle error
            }
            console.log('Send Storage Error!');

        }
});
}
```

CODE 13. Sending from storage

Storing single resulting report JSON offline is quite easy as you only need to serialize it and save it in the local storage as a string. Storing multiple reports is more complicated, as there are other reports that must not be overwritten.

```
function saveReportToStorage(json){
        if(getData("reportCell") == null){
                var saveArr = new Array();
                saveArr.push(json);
        }else{
                var saveArr = JSON.parse(getData("reportCell"));
                saveArr.push(json);
        }

        setData("reportCell",JSON.stringify(saveArr));
        callPopup("offlineReportAttempt");

}
```

CODE 14. Storing report

5 CONCLUSION

In my thesis I tried going through my experience of the creation process of a cross-platform hybrid application. I have mostly focused on the usability aspect of the application and selection of tool. I have skipped explaining basic coding explaination as there was so many different libraries, frameworks and tools that covering basic usage for each of them is a serious overkill and falls out of the scope of my thesis.

UX/UI for application such as NearMiss is a critically important aspect which defines the application itself. If application is designed properly it can make the whole process convenient and pleasant for users. On the other hand, poorly thought through application can make the task it tries to simplify even more tedious and hard. Developers need to clearly understand their target audience and focus on providing users with the best possible experience. In enterprise sector UX is given a very low priority since people that actually make the decisions are only interested in the functional aspect of the product. I find it unfair that in the end they won't be the ones using it. I have met a great number of people who work with software they hate, but are presented with no other option by their employer.

I also had an invaluable experience working with cross-platform development. I would like to mention that it have grown tremendeously over the years to be a great option and a competitor to native development. Many developers will find it very efficient and attractive to create reusable code that can later be used for the multiple platforms. This approach can save a lot of time as there is no need to study each platform and and their API individually. I find it great that there is so much flexibility in the selection of approaches and tools available on the market today. Such diversity can attract more people to study and work in the field which will inevitably lead to the appearance of new applicable techniques and further increase number of options for everyone's benefit. Currently there are some downsides and limitation that developers face during application development but it is still viable approach and it will only get better in recent future. It is hard to ignore that the the technology evolves rapidly and the shared code percentage across platforms will gradually rise as well as the quality of such applications.

BIBLIOGRAPHY

Vandersluis, Chris 2013. Enterprise system best practices. WWW-Document. https://support.office.com/en-us/article/Enterprise-system-best-practices-white-paper-cf300560-ed31-4db8-9e53-b5a1003ea26e. Referred 22.09.2016.

Oracle, 2015. Java timeline. WWW-Document http://oracle.com.edgesuite.net/timeline/java/. Referred 22.09.2016.

Oliver, Andrew 2013. Love and hate for Java 8. WWW-Document. http://www.infoworld.com/article/2611558/application-development/love-and-hate-for-java-8.html. Updated 25.06.2013. Referred 22.09.2016.

Hoon Park, Se 2012. Understanding JVM Internals. WWW-Document. http://www.cubrid.org/blog/dev-platform/understanding-jvm-internals/. Referred 22.09.2016.

Perry, Mike and Oskov, Nasko. Introduction to Reverse Engineering Software. E-Book. http://althing.cs.dartmouth.edu/local/www.acm.uiuc.edu/sigmil/RevEng/ch02.html. Updated 28.06.2006. Referred 23.09.2016

W3 Foundation 2010. HTML5 Reference. WWW-Document. https://dev.w3.org/html5/html-author. Updated 09.08.2010. Referred 24.09.2016

Way, Jeffrey. 28 HTML5 Features, Tips and Techniques You Must Know. WWW-Document. https://code.tutsplus.com/tutorials/28-html5-features-tips-and-techniques-you-must-know--net-13520. Updated 14.06.2011. Referred 24.09.2016

W3Schools. CSS Reference. WWW-Document. http://www.w3schools.com/cssref/. Updated 2015. Referred 25.09.2016.

Angelov, Martin. 12 Awesome CSS3 Features That You Can Finally Start Using. WWW-Document. http://tutorialzine.com/2013/10/12-awesome-css3-features-you-can-finally-use/. Updated 25.10.2013. Referred 25.09.2016

W3.org. Short history of JavaScript. WWW-Document https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript. Updated 27.07.2012. Referred 26.09.2016

Champeon, Steve 2001. JavaScript: How Did We Get Here? WWW-Document. http://archive.oreilly.com/pub/a/javascript/2001/04/06/js_history.html. Updated 06.04.2001. Referred 26.09.2016.

Eich, Brendan 2011. Harmony Of My Dreams. WWW-Document. https://brendaneich.com/2011/01/harmony-of-my-dreams/. Updated 01.2011. Referred 26.09.2016.

Salsita Software 2016. Top rated libraries. WWW-Document. https://www.javascripting.com/?sort=rating. Updated 26.09.2016. Referred 26.09.2016.

Toubassi, Garrick 2014. Official Gmail Blog. WWW-Document. https://gmail.googleblog.com/2014/11/going-under-hood-of-inbox.html. Updated 20.11.2014. Referred 02.10.2016.

Google Web Toolkit 2016. Documentation. WWW-Document. http://www.gwtproject.org/doc/. Updated 2016. Referred 02.10.2016.

Neethling, Schalk 2008. Understanding GWT Compiler. WWW-Document. https://dzone.com/articles/understanding-gwt-compiler. Updated 17.08.2008. Referred 02.10.2016.

Buzdin, Dmitry 2011. GWT Compiler Optimizations. WWW-Document. http://www.buzdin.lv/2011/02/gwt-compiler-optimizations.html. Updated 16.02.2011. Referred 02.10.2016.

Cordova 2016. Architectural Overview of Cordova platform. WWW-Document. https://cordova.apache.org/docs/en/latest/guide/overview. Updated 23.04.2016. Referred 04.02.2016.

Reineke, Adam. Optimize the performance of a Cordova app. WWW-Document https://taco.visualstudio.com/en-us/docs/better-web-performance/. Updated 08.10.2015. Referred 04.02.2016.