CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Computer Graphics and Interaction

Accessible multi-platform navigation application for navigation of blind pedestrians

Bachelor‘s thesis

Study programme: **Software Engineering and Management**
Field of study: **Web and Multimedia**

Supervisor: **Jan Balata**

Vojtěch Gintner

May 2017 in Prague

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Vojtěch Gintner

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Přístupná multiplatformní mobilní aplikace pro navigaci nevidomých chodců

Pokyny pro vypracování:

Navrhněte a implementujte multiplatformní aplikaci pro mobilní telefon pro navigaci nevidomých chodců.

1. Seznamte se aplikačním rozhraním navigační služby pro nevidomé Naviterier.
2. Proveďte analýzu nástrojů pro multiplatformní vývoj na mobilních zařízeních, porovnejte přístupnost výsledných aplikací a vhodný nástroj vyberte.
3. Vytvořte high-fidelity prototyp navigační aplikace, proveďte a vyhodnoťte metody testování použitelnosti bez uživatelů (kognitivní průchod, heuristická analýza, expertní evaluace).
4. Na základě výsledků implementujte multiplatformní aplikaci pro mobilní telefon.
5. Proveďte a vyhodnoťte test použitelnosti implementované aplikace v exteriéru alespoň s šesti participanty z cílové skupiny.

Zaměřte se především na pokrytí velkého množství mobilních platforem včetně starších zařízení, lokalizaci na úrovni chodníků, hlášení problémů uživatelů na trase, komunitní sběr informací a vyhledávání cílových bodů.

Seznam odborné literatury:

1. Craig, James, et al. 'Accessible Rich Internet Applications (WAI-ARIA) 1.0. 2009.' Online: http://www.w3.org/TR/wai-aria/ on 2016-10-25.
2. Špinar, David. Tvoříme přístupné webové stránky: připraveno s ohledem na novelu Zákona č. 365/2000 Sb., o informačních systémech veřejné správy. Zoner Press, 2004.
3. Power, Christopher, et al. 'Guidelines are only half of the story: accessibility problems encountered by blind users on the web.' Proceedings of the SIGCHI conference on human factors in computing systems. ACM, 2012.

Vedoucí: Ing. Jan Balata

Platnost zadání: do konce zimního semestru 2018/2019

L.S.

prof. Ing Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka CSc.
děkan

V Praze dne 4.4.2017

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used in compliance with Methodological Guidelines on Compliance with Ethical Principles in the Preparation of Graduate Thesis (1/2009). I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on May 26, 2017 …….

Acknowledgements

I take this opportunity to express gratitude to Ing. Jan Balata, my supervisor, for his help and valuable guidance and doc. Ing. Zděněk Míkovec Ph.D. for his assistance with getting participants for testing. I also want to gratefully thank my family, girlfriend and friends for their support, attention and helping me finish this.

Abstract

The purpose of this bachelor's thesis is to design and implement accessible multi-platform mobile navigation application for navigation of visually-impaired pedestrians. I started with stating problems and goals that were necessary to achieve. After that I compared existing navigation applications for sighted and visually-impaired users, platforms used by visually-impaired, accessibility of these platforms and available tools for application creation. At the beginning of the design process were modelled particular use cases and corresponding storyboards. Coming next were created paper, other low-fidelity and high-fidelity prototypes one after another. User Centered Design methodology was employed during the whole design process. Every iteration were consulted with one sighted and one visually-impaired expert. The final application was sucessfully evaluated with 6 participants in the streets.

Key words

multi-platform application, accessibility, visually-impaired users, navigation

Abstrakt

Cílem této bakalářské práce je navrhnout a implementovat přístupnou, multiplatformní, mobilní, navigační aplikaci pro navigaci zrakově postižených chodců. Po stanovení problémů a cílů, kterých bylo třeba dosáhnout, jsem porovnal existující navigační aplikace pro vidomé i nevidomé, platformy používané nevidomými, přístupnost jednotlivých platforem a dostupné nástroje pro tvorbu aplikací. Aplikace byla navržena metodou User Centered Design. Byly namodelovány jednotlivé případy užití a k nim příslušné storyboardy. Postupně byly vytvořeny papírové návrhy, low-fidelity prototypy a high-fidelity prototypy. Jednotlivé iterace byly konzultovány s vidomým a nevidomým expertem. Výsledná aplikace byla úspěšně otestována v terénu se 6 nevidomými participanty.

Klíčová slova

multiplatformní aplikace, přístupnost, nevidomí uživatelé, navigace

Table of contents

Table of figures

Table of tables

1 Introduction

Navigation of blind and visually-impaired people using a mobile device is a challenging task. There are some applications that try to do that in some extent and you can read about it later in next chapter. But there is a team from the Department of Computer Graphics and Interaction at Czech Technical University that works on exactly this kind of project. The project is named **Naviterier**[1]. You can read more about it on the website but simply it is a database of street topology like Google Maps[2] or Mapy.cz by Seznam[3] have. The first difference is that Naviterier registers each pavement by itself, not just the whole street. In addition to the pavements there is also an incredible amount of detailed data about the pavement width, pedestrian crossings and their types, traffic direction, pavement surface and pavement slope, corner roudness and many more details. All this data can be then transformed in very verbose description of a route looking like it was written by a real person or a friend just for the particular route and particular blind user. Currently is covered about 150 kilometers of pavements in the city centre of Prague around Charles' Square.

There is an API that takes point A and point B and creates detailed itinerary of the route between those points if all the pavements between are fully covered by the data. I joined the team last year and my first task there was to create fully accessible website which could take the start and the destination points and return corresponding itinerary of the route. I used PHP for creating the **Naviterier Routeplanner**[4] which you can try on the website. The Routeplanner is also responsible and thus usable on the desktop as well as almost any mobile device including Symbian based devices. Work on that website continued and I integrated searching of points of interest in addition to addresses as well as typing error correction. Google places API and Google geocoding API were used in conjunction with the Naviterier database to achieve these functions.

All the experience I gathered when designing and implementing the Routeplanner as well as data background of Naviterier will be used to achieve the best results. Most of the address and points of interest searching and route retrieving was previously implemented by me in a form of REST API [1] which I will use in the navigation application to retrieve the data.

I focus on providing an accessible mobile user interface for existing navigation system Naviterier. Thanks to that, I don't need to spend thousands of hours inventing complex navigation algorithms and I can simply focus on providing flawless user interface and pay attention to so important details. Perfectly and precisely designed user interface is the key to create a fully functional navigation system that is not only user-friendly and easy to use, but also a solid support of visually-impaired on their way through unkown paths. During all designing and prototyping phases, I employ of **User-Centered Design methodology** [2]. In my design, I focus on sidewalk level localization accuracy using conversation with the blind user and crowdsourcing additional information about their whereabouts.

---

[1] www.naviterier.cz
[2] maps.google.com
[3] mapy.cz
[4] www.naviterier.cz/livedemo/

2 Motivation and related work

I compared similar applications on the market and I found out that currently available navigation applications for visually-impaired are inadequate. Mostly based on the same data that is being used for navigating sighted users. Basically none of the applications can provide safe navigation on the streets as these applications don't mind pedestrian crossings and other important obstacles.

In order to overcome other navigation applications I need to state a few things first that should this application take care of. Safe mobile navigation of visually-impaired pedestrians consists of 3 separate problems.

The first one is providing user sufficient, detailed information about the route that is up to date. Being detailed and accurate as possible is a must. Naviterier provides very detailed data like corner roundness, pedestrian crossings, slope, material of the pavement, etc. so this is taken care of. The application will act as eyes and sense of navigation for the user. World around us is constantly changing. How can we be sure the data in database is up to date? I will include additional crowdsourcing for gathering not only new data but mostly for check and update of current data in Naviterier database.

The second problem mostly belongs to the start of navigation. Not everytime you know your exact address or place you are standing at. Not even as sighted user does. This means I need to include GPS localization to determine user's current location as an alternative to inputting start manually.

The third problem is not less important than previous two. What if user gets into a trouble? I need to provide a way for the user to recognize, diagnose and recover from any problem on the route. That means I'll include problem reporting, that will announce new problem on the route to Naviterier database and also gives user instruction on how to recover from the problem. User should be also able to check his/her position to know if he/she is on the route or went off.

I also compared mobile platforms that blind users use, accessibility across platforms, native and multi-platform APIs, framework options and then justified my choices.

2.1 Already existing navigation applications

Usual navigation applications for sighted user would be **Google Maps**[5], **Maps by Apple**[6] or **Mapy.cz by Seznam**[7]. Mapy.cz does not provide currently navigation for visually impaired. Google and Apple does. Both navigation applications are "turn by turn" type and work very similarly. They use users current location and split the route by each street intersection. The user can see current instruction in form of "Go X meters and then turn slightly Y on street Z". Neither of these applications take into account pavements, pedestrian crossings, stairs or any

[5] maps.google.com
[6] apple.com/ios/maps/
[7] mapy.cz

other essential information about the route. Maps by Apple also allow to move a finger on the screen reading aloud name of any street it touches including its orientation on the map. Unfortunately none of the applications can be safely used for navigation of blind pedestrian because the amount of information given to the user is quite vague and inadequate.

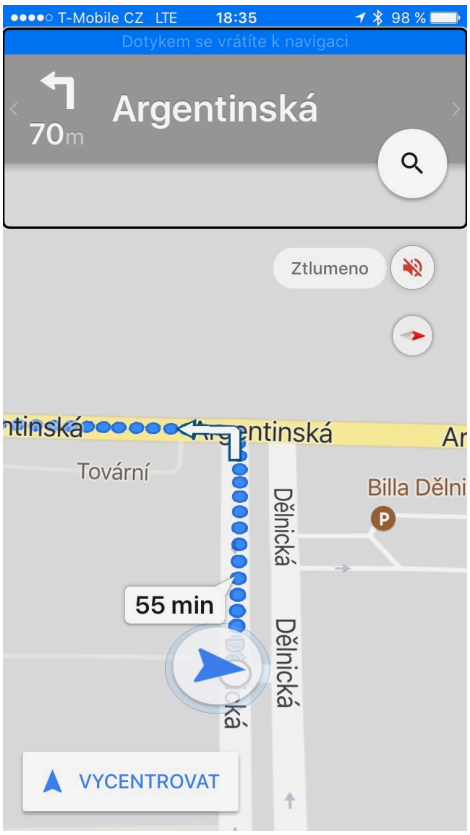


Figure 2.1: Google Maps user interface

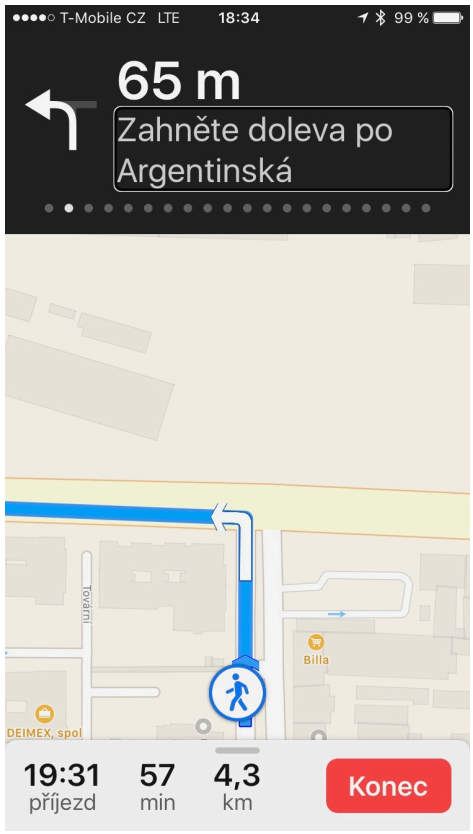


Figure 2.2: Maps by Apple user interface

Fortunately there are 4 established navigation applications that are blindness-aware and especially made for visually-impaired users.

These applications are:

- **BlindSquare** (iOS) - blindsquare.com
- **Nearby Explorer** (Android, iOS) - tech.aph.org/ne/index.html
- **The Seeing Eye GPS** (iOS) - itunes.apple.com/app/id668624446
- **Lazarillo GPS for Blind** (Android, iOS) - lazarillo.cl

I need to state at first that none of these is a fully fledged navigation application in terms of step by step navigating from point A to point B like Google Maps or Maps by Apple try to achieve. All of the listed applications provide basically the same functionality. These applications keep track of your current location and announce you information about your current location, heading, distance to your destination, upcoming street intersections and **nearby surroundings** like coffee shops, restaurants, banks and other points of interest. None of these unfortunately provide any detailed information about pavements or pedestrian crossings. All of the listed applications are designed to act as a complement to Google Maps or Maps by Apple.

All of them excluding Lazarillo GPS for Blind are payed applications, either by one time purchase or monthly plan. BlindSquare is probably the most popular, it uses Foursquare and Open Street Map as a data source. The Seeing Eye GPS uses Google Maps, Foursquare or TomTom as a data source. Data source of Lazarillo GPS for Blind stays unknown to me. Downside of all three applications is a need to stay connected to the internet all the time. Nearby Explorer compensates this disadvantage by downloading data from Google Places offline to your device so it can be used without internet connection later on. Unfortunately only a few cities in the United States and Canada are currently fully supported by Nearby Explorer.

The last application I want to mention is not designed to be a navigation application or anything realted at all, but it can get handy when a blind user needs to catch a direction.
This application is called **Remote Assistant** and it's availbale for iOS. The applications provides a support service for blind people by using GPS location and camera of their smartphone. How it works? It's quite simple. Blind user needs someone's pair of eyes to describe him/her what's in front. The user launches that application and the applications starts a call. Video stream and current location is then sent to the operator who can respond with a description what he/she sees through the camera. It can be used at home if you need help with reading something like a manual or buttons on your washing machine as well as outdoors when you need to "look around" and catch the right direction or spot that restaurant across the street.

2.2 Mobile platforms used by blind users

Although the main focus is on modern Android and iOS devices, as of 2016 research done by J. Balata and Z. Mikovec in Czech republic [3] reveals that there is still many people using older mobile operating system **Symbian** (62% of 25 people). Android is on second place with 21% and iOS on the third place with 17%. Although the trend is towards Symbian percentage decrease in the following years, it should be considered in approach for the new navigation application to support also Symbian devices.

Figure 2.3 displays percentage of devices with a particular operating system within the group of respondents (solid fill) (n = 25; 29 devices) and the predicted percentage of devices with a particular operating system in near future (dashed fill)

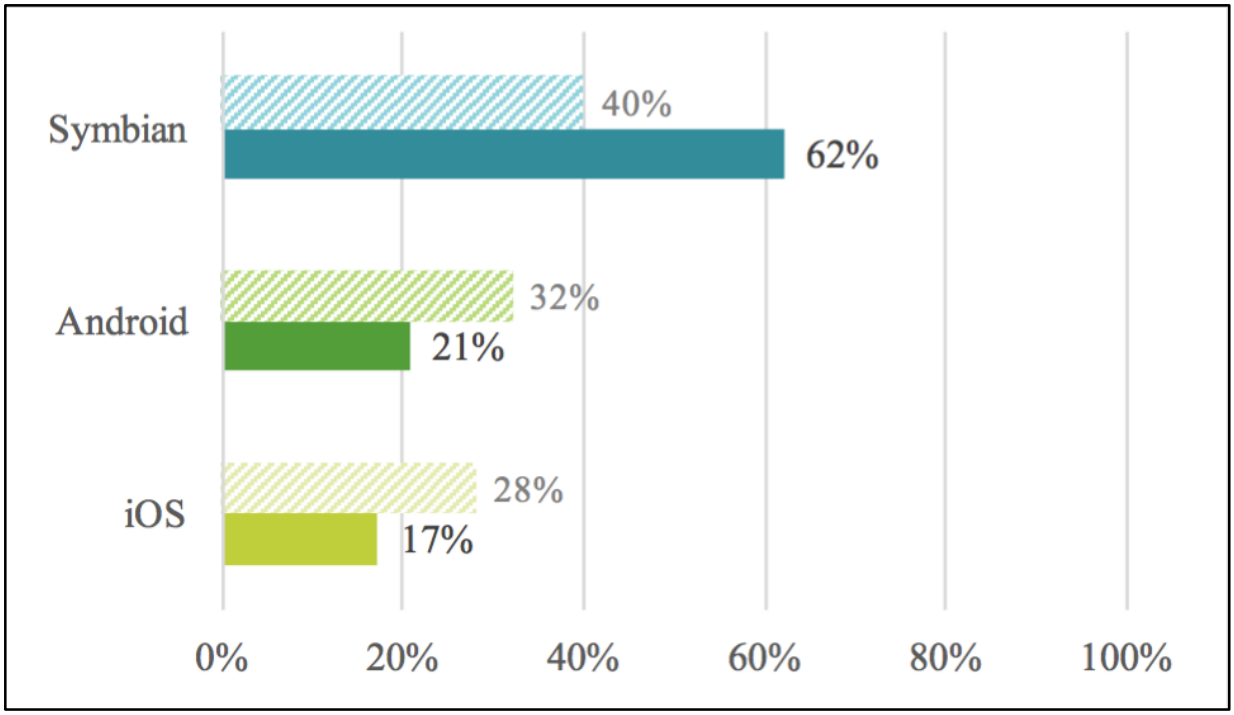


Figure 2.3: Percentage of devices with a particular operating system

2.3 Platforms accessibility comparison

All systems including desktop offer screen readers and accessibility API capable of running a navigation application. There are slight differences across platforms though. The most equal accessibility behavior can be recorded in web browsers across all platforms. Screen reader coverage is displayed in table 2-1.

Operating system	Native screen reader	Third-party screen reader
Windows	Narrator (basic)	JAWS, NVDA
Linux		Orca, BRLTTY
Mac OS	VoiceOver	
Symbian		Talks, Mobile Speak
Android	Talkback	
iOS	VoiceOver	

Table 2-1: Operating systems and their screen readers

2.4 Native vs web-based

Naviterier Routeplanner currently works as a standalone website or let's say a web application. Key advantage of that is being multi-platform, unified and easy to maintain or update, because no installation is needed. That brings a question if web-based approach is usable for the kind of application I am going to create.

Both native and web offer a rich variety of similar accessibility API.

Native accessibility API is very rich and offers a wide variety of options and possibilities but is very **diverse** across platforms and usually works differently.

Web API fulfills whole **WAI-ARIA specification** [4]. It is quite more **universal** and little poorer than native, but with composition of multiple options can developer achieve very similar level of control in web environment thus achieve the same results. Web-based application has an advantage of being multi-platform and possibly unified across devices.

An application needs to respond to various user actions or system events. That is being achieved by driving screen reader focus to corresponding elements. Either **synchronously** as a direct response to user action or **asynchrounously** as a response to a system event.

Maintaining screen reader **element focus** through asynchronous events is little easier in native toolset than on the web, especially with iOS, because asynchronous manipulation of screen reader focus is very restricted in iOS 10 and in newer versions expected. However, web toolset offers aria live regions[8] as well as aria role alert[9] for asynchrounous actions and Document Object Model (DOM)[10] element focus() method[11] to force focus on an element in response to a synchronous action to compensate. That is one of the largest differences between native and web accessibiity APIs. Native accessibility API is not as restrictive as web API.

Native pros (+)

- Nothing beats performance of native application, multithreading enabled.
- More situationally specific accessibility API, a lot of options and possibilities for the same solution.
- Easy access to GPS, compass, phone storage and other device resources.

Native cons (-)

- Totally different toolsets and diverse accessibility API. Results in various user experience on different operating systems.
- Necessity to develop and maintain multiple distinct applications.
- No desktop usage.

Web pros (+)

- Uniform toolset for a large number of devices with almost equally behaving accessibility APIs.
- Only one application to develop and maintain for multiple platforms. Easy and controlled distribution of application and its updates
- Possible desktop or reader usage.
- Option to seamlessly transform into hybrid application and get best of both worlds.

Web cons (-)

- Running in browser or browser wrapper.
- Only web specific elements can be used.
- No multithreading allowed. Performance is reduced.
- Restricted access to Geolocation API and device resources in browser.

[8] w3.org/TR/wai-aria/states_and_properties#aria-live
[9] w3.org/TR/wai-aria/roles#alert
[10] w3.org/TR/WD-DOM/introduction.html
[11] w3.org/TR/2016/REC-html51-20161101/editing.html#focus-management-apis

2.5 Android vs iOS

Screen readers on Android and iOS behave very similarly with one key difference. **Every web browser on Android can behave and behaves differently** because each browser adapts its own version of accessibiity API of which is currently most stable FireFox. While **on iOS devices can users experience the same behavior in any web browser**. All web browsers on iOS devices are restricted to stick with one particular accessibility API adapted by Apple for Safari. These remarks and many more comparison were done by Paul J. Adams *„iOS vs. Android Accessibility"* [5]

2.6 Website and web application accessibility

In 1999 the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C) published Web Content Accessibility Guidelines (**WCAG 1.0**) [6] which have been replaced in 2008 by **WCAG 2.0** [7]. These are guidelines for website developers to ensure maximum accessibility across the web especially for visually-impaired users. With growing amount of asynchronous XMLHttpRequests[12] and adoption of mobile application grew disunity of accessibility through internet applications across platforms. Those events gave birth to Web Accessibility Initiative - Accessible Rich Internet Applications (**WAI-ARIA**) 1.0 [4]. These are set of rules, and utilities for websites and applications taming structures and behaviour of dynamic applications to be accessibility friendly.

As shown in a paper Guidelines are Only Half of the Story [8]. None of the current guidelines or set of rules can fully ensure exceptional accessibility based on headless following of guidelines and automated testing. That's a good reason to employ User
-Centered Design [2] methodlogy. Proper **user evaluation** is unavoidable to achieve fully usable and accessible user interface. Therefore session with blind or any other visually-impaired subjects is necessary for maximum results. Although some usual problems belong more likely to user experience (UX) it's not true that mentioned guidelines make a good design by itself without user testing.

2.7 Hybrid application

Advantage of web applications is the opportunity to transform them into hybrid applications. With just a little additional effort can developer turn any web application into almost **native alike** with usage of tools for hybrid application conversion and get the **best of both worlds**. These tools basically take the web application and wrap it in a thin layer of code called **web wrapper**. That enables the application to be installed and treated like any other native application, including native access to Compass, GPS, Camera and other mobile resource APIs. Native alike applications for multiple platforms can be created and maintained with just a single HTML & JavaScript web application.

Currently the best of the hybrid transforming tools is **Cordova** by Apache[13]. Transformation for individual platforms is a matter of seconds within provided command line tools. Between

[12] xhr.spec.whatwg.org
[13] cordova.apache.org

supported platforms belongs Android, iOS, Windows Phone, Blackberry OS, Ubuntu, Firefox OS, FireOS, LGwebOS and web browser.

2.8 With or without JavaScript

Going without JavaScript opens up doors to extremely old or lightweight devices. It also makes accessibility API a little more powerful as there is no dynamic loading content and screen reader determines when to read out on its own. On the other hand, it is very limitting in terms of what you can really create. It removes the possibility to use **Geolocation API**, which should be the power source of the application. Using JavaScript brings more life and speed for the application together with powerful tools including dynamic responding to user interaction. With REST API on back-end makes JavaScript necessary instrument in today's world of web and mobile applications.

2.9 Symbian

Although symbian devices are widely used between visually-impaired people, no hybrid nor web based application can guarantee 100% functionality. Symbian devices are currently outdated and cannot make use of all modern tools. A lot of them doesn't even have a GPS support. There is an option to select supported and tested devices that will work with the application by hand or developing native application for specific Symbian based devices. Developing native application for Symbian devices would take a lot of additional time. And also, that would mean it's not really multi-platform application. Because of these reasons I decide to not support Symbian devices at the moment. As a compensation can any Symbian user use Routeplanner instead of a standalone application. Or receive a distribution of the hybrid application only if the device specification fits the requirements for the application and will be successfully tested.

2.10 JavaScript frameworks

Accessible web application can be developed in different possible ways. Either using an already existing framework or library or creating from scratch. Using a framework or library has the advantage of fast and reliable application with easy to manage source code, but can bring restrictions especially for the accessibility. Building from scratch sets developer free in the terms of accessibility. Unfortunately creating everything from scratch means much slower development and harder maintenance. I should consider the path that will the developer go while designing accessible web application. It can prevent encountering problems later on that some feature is not possible to create within the chosen frameworks and libraries.

My considerable options are:

No framework or library, plain HTML and JavaScript. Probably the messiest way possible. No modularity, multiple approaches mixed and almost impossible long term maintaining especially if working in a team in the future. Most of the time gets developer into inventing what is already invented.

jQuery[14] and other related DOM-manipulative frameworks and libraries are very strong helpers if used rarely. Otherwise, it brings more chaos than benefit while overly excessive DOM manipulation definitely isn't accessibility-friendly neither. jQuery excessively manipulates the DOM and therefore we can expect undefined behavior of the screen reader. Building the whole application on jQuery is a also a nonsense move in terms of future maintaining.

Meteor[15] is a full-stack JavaScript framework allowing to build whole applications using only JavaScript. That means Front-end and Back-end are both written bound together hand in hand in one single language and framework. There is an extensive support to live updates and event-driven actions. The whole framework is quite opinionated and forces developers to use pre-made plugins and modules including Back-end with MongoDB. Bootstrapping an application is on the other hand very fast and easy if it fits into the set of available plugins and modules. **Blaze**[16] is the default templating engine, but Meteor also supports use of Angular (directives) or React (components) templating. Unfortunately this quite closed ecosystem makes it unnecessarily complicated to implement all the WAI-ARIA [4] rules a specifications.

Ember.js[17] covers up most of the Front-end tools needed. That means the application logic and view layers as well as routing and asynchronous data loading from a server. It uses **Handlebars**[18] templating system supporting 2-way data binding. Ember is one of the more opinionated frameworks that induces use of specific tools and techniques. It suits well for most of the common applications, but falls little short in more custom builds. Although handlebars can be accessible and there are some minor accessibility plugins, there is no full accessibility support.

AngularJS[19] (v1.x.x) is the most common framework these days (2014 - 2017+). It offers whole variety of tools starting with very powerful view layer supporting 2-way data binding including custom made "DOM elements" called directives for excellent reusability. Controllers for lightweight functionality, logic behind the scenes and Services to take care of the communication with server or just simple routing. Angular excels at easy binding and dependency injection. It also allows use of jQuery or similar DOM-manipulative libraries along with the view layer. Although it is slightly opinionated its largest disadvantage is in fact in the quite liberal approach to achieving tasks where a single goal can be done in numerous absolutely different ways contributing to inconsistency. The community is huge and a lot of help and pre-made solutions or modules can be found easily. Angular offers good accessibility support of it's own, but there is also a lot of community made modules that enhance the already great support of accessibility.

[14] jquery.com
[15] meteor.com
[16] blazejs.org
[17] emberjs.com
[18] handlebarsjs.com
[19] angularjs.org

React[20] belongs into view layer only libraries. It is quite different because of its suggested rules of development namely one-way data binding and uni-directional data flow. Recommended immutability, global and local application states. Thanks to a virtual DOM excels React at exceptional performance. All DOM manipulation happens in virtual DOM and only the final version gets out into real DOM. This saves a lot of computing power, because DOM mutations and redrawing are the slowest operations by far. Developer must provide application logic (self-made or a framework) in order to work properly. Recommended logical libraries are state managers like **Flux**[21] or **Redux**[22]. Components are the main building block of React. Similar to directives with extended properties and functionality. React also brings **JSX**[23]. That's JavaScript mixed with HTML templating together into single file. For me as a developer is React very familiar one. Regular HTML usage + simple JavaScript event binding offers great accessibility alongside that extreme performance.

Backbone.js[24] (+ **Underscore.js**[25]) Is simply a basic and lightweight spine for any application binding data and templates into distinct layers. There is no support for 2-way binding and mutations happen right in the DOM possibly leading to quite fragile application design easily exposed to memory leaks and performance drops. The whole framework is quite liberal and allows a lot of adaptation and that is a double-edged blade. Backbone provides amazing data management thanks to Underscore. The data management is native-alike and contains one of the best collections and objects related set of methods and functions. Because Backbone.js is focusing on data management, accessibiity is determined by the templating system used.

Vue.js[26] like React is a view layer library very similar to Angular's view layer. It supports 2-way data binding and all template variables are fully reactive. The main building block is directives as well. Used with a good data managing framework can provide good accessibility. But for example that 2-way databinding is quite unaccessible by itself.

2.11 Goals

I'll list there the goals I want to achieve with the application.

- Designing flawless accessible user interface that is very simple and does not need to be learnt.
- Creating multi-platform application that can be used on as many devices as possible.
- Developing safe and reliable application that is going to be used on regular basis.
- Testing the application with blind pedestrians in real life conditions ensuring usability on the street without any further assistence.
- Continue work on the application after finishing bachelor's thesis. Having a long-term plan for maintenance and upgrades for the application.

[20] facebook.github.io/react/
[21] facebook.github.io/flux/
[22] redux.js.org
[23] facebook.github.io/react/docs/jsx-in-depth.html
[24] backbonejs.org
[25] underscorejs.org
[26] vuejs.org

2.12 Chosen path

Web-based approach was chosen for the Naviterier application because of its **multi-platform** use, **easy future maintenance** and sustainability in addition to the possibility of transforming into a **hybrid application**. All attributes of web-based application match my needs and goals with an advantage of faster development.

Because all our target devices are cabale of using JavaScript and we need to use Geolocation API in order to localize the user, it has been decided to go with the JavaScript based option.

Simple **extensibility** and **easy maintaining** were one of the goals of Naviterier application and therefore going without a framework/library or using DOM-modulative library is not an option. As the data structure is simple there is no need for a data library like Backbone at the moment. It doesn't fit the needs because of its DOM-modulative traits too.

Although Meteor and Ember bring a lot to the table, their opinionated approach is quite limiting for such a unique project. A great view layer and very lightweight data model is clearly the choice here. Because I have way more experience with React and I tend to use uni-directional data flow a lot (namely Redux). **I chose React** over Angular or Vue as the library to go with. Another reason would be lack of uniformity in solutions on the Angular side. Or need to add some more complex data structure for Vue. Advantages of using React is a huge growing fan crowd that creates a lot of interesting modules that could be used in the project. Facebook company backing the whole React project with a lot of work spending each day on maintenance. Simple modularity. And also that almost forced uni-directional flow that in my opinion prevents a lot of errors and bad habits just by design.

3 Design and low-fidelity prototyping

After my research I have all the information I need to start designing the application. I will analyse all problems and then create use cases that will be transformed into low-fidelity prototypes. Low-fidelity prototypes can be quickly sketched or coded and so will be also perfect for early evaluation.

During the designing and prototyping I used a book "*Tvoříme přístupné webové stránky*" by David Špinar [9] as a guide in addition to other sources such as Mozilla Developer Network[27].

3.1 Expected functionality

At first I need to specify expected functionality of the application in order to start designing. These are the most important things that the application should contain.

3.1.1 GPS localization

GPS localization makes the application really usable, so it's a must-have. It would be just another Routeplanner without it. GPS should be usable for finding the location from which the user starts its route and also to check if the user is on the route or not. It may be fine to have continuos announcing of current location, surroundings, distance to destination and warning when walking off the route. But there is many other applications out there that do just this. This application will purely focus on navigation, not distraction. And even if I wanted to integrate this. Unfortunately there is no background ready for continual localization at Naviterier. So it's not possible with the data right now. In near future most likely, but not now. In addition, continual using of GPS draws a lot of power and batteries drain so fast. Blind pedestrian in the middle of unknown without assistance and dead phone battery is not a great situation too.

3.1.2 Comprehensive description of the route

Segmented route with as much details as possible will be a key feature of this application. Just like in the Routeplanner. The same data, the same style, the same approach.

3.1.3 Crowdsourcing

In order to keep the data about routes up to date and even to add new details into the database, there is a need to regularly ask users short questions to fill in missing or outdated information about the route and surroundings. This crowdsourcing shouldn’t be disturbing but still frequent enough to validate as much data with as much users as possible.

3.1.4 Problem reporting

The city is changing all the time, building sites, new pavements, closed streets and redirected pathways. Route data can expire any time and user can get confused or even lost. The application needs to provide a way for the user to report anything unexpected or suspicious

[27] http://developer.mozilla.org/en-US/docs/Learn/Accessibility/What_is_accessibility

including when the user gets lost for some reason. The problem should be reported for further validation and user must receive directions what to do or which alternative path to go.

3.1.5 Location check

In the case of user feeling like being lost or not sure if going right. The application should provide an option to check user's current GPS location according to the route. User should receive clear information if he/she goes right and if not, how much is the user off the route and what to do to get back.

3.2 Problems

As of now, there is currently only one known problem I'll need to overcome somehow.

3.2.1 GPS accuracy and precision

We have all been in a situation when GPS points to a place that is far away or within a large radius. Let's state it clear. Do you know the difference between accuracy and precision? I include an illustration to be on the safe side.

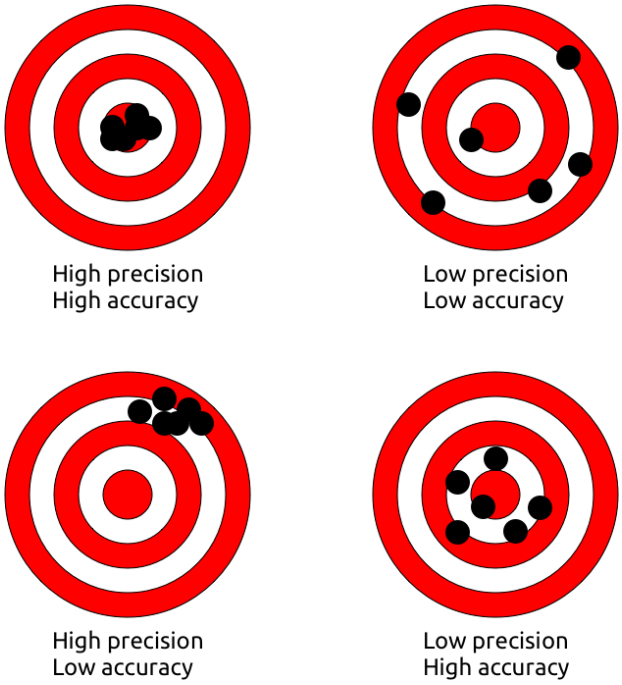


Figure 3.1: Difference between accuracy and precision

The problem with GPS is that it's usually quite not precise. And in the cities not even accurate. I'll explain this. When you are trying to localize yourself using some map application. You usually see your predicted position on the map with a fade radius around that point. This is the precision part. It's a radius of expected possible margin of error. It can be anywhere from a few meters to kilometers. It usually starts quite high, even hundreds of meters and get to about 16 - 40 meters in a few seconds of GPS turned on. You can make it get even lower to about 4-8 meters just by moving or having a strong signal to as many satellites as possible. Unfortunately this all is a product of internal predicting and mystery estimations. It's not just GPS but even cell phone signal and names of nearby wifi networks taken in account of refining

the precision. I won't go into details on that as I am no expert in these fields. All I know is that GPS is not precise enough to determine on which side of a road is the user standing. That's unfortunately crucial for this application. But it's going to get even worse.

In the dense city with a lot of tall buildings can be GPS signal not even shielded off, but even bounced off a building resulting in the second type of a problem and that is low accuracy. Even if the precision radius would be half a meter all it takes is just one building acting like a shiny mirror for the GPS signal. Signal bounces, but the device doesn't know that. All the calculations are done right, but still the location can seem to be even 60 meters off. Usually right in the middle of the building or block of buildings.

That's not one, but two problems with GPS. How can I trust the GPS when it seems so untrustworthy? Luckily there is one option available. In order to detect the user location with pavement level of accuracy and precision and point it the right way, we don’t only need really precise location and direction at which is the user turned, but also to know on which side of the road the user is.

We can only know on which side of the road the user is by asking the user itself. That's not the cleanest resolution of the problem. But without totally precise and accurate GPS, we can't trust it.

What about the direction that is user facing? We can quite trust the compass in smartphones. But can we trust that user? Sighted users will hold their phone with the display facing them and thus data reading from the compass is just right. Blind user on a noisy street can hold the phone up to an ear, facing sideways. Blind user will probably hold a white cane and can be in a stressful situation. I can't rely on the compass neither.

So let's sum it up. We may know on which side of the road the user is. But we don't know the exact location and not even direction the user is facing. Is there anything to fix this? Yes!

When device moves, GPS get's more precise. In addition you can collect all received locations of the user while moving. You can easily determine the direction from the movement. So now is just that exact location last problem remaining. Fortunately Naviterier can take quite imprecise point with a direction and side on which is a road, compare all surrounding pavements and respond with an array of most likely matching results sorted by distance from the current estimated location.

So to receive the best results, all we have to do is to kindly ask the user to walk a short distance in a straight line to refine the user location. Only question is, how short distance can that be. But that will reveal itself with implementation and first testing in exterior.

3.3 Use cases and storyboards

The first step will be creating proper use cases of all possible actions that users may take. Use cases were later transformed into simple storyboards that ensure easier representation. These use cases will be used for establishing all needed application screens and functions.

3.3.1 Localization

- User asks for localization while searching route between start and destination location.
- System tries to get location, asks user to walk a few meters in order to increase GPS precision while leaving the phone screen on.
- System fails to receive user location - tells user about inaccuracy of GPS and asks to either input location or walk until system receives user location.
- OR
- System receives user location - system asks on which side of the road is user.

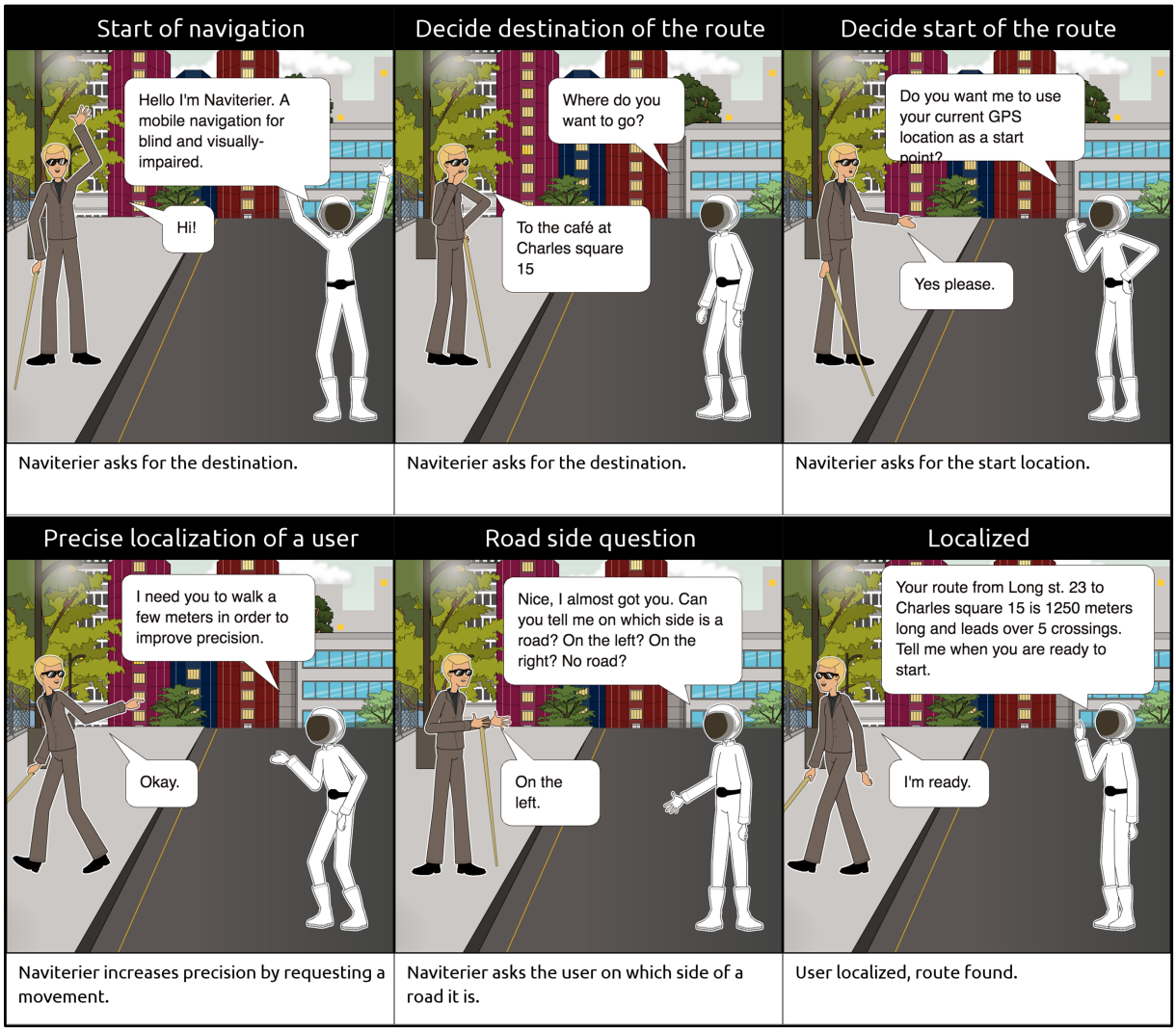


Figure 3.2: Storyboard – Localization and start of navigation

3.3.2 Navigation

- System navigates user on the route based on start and destination location.
- User continues navigation step by step on its own.
- User thinks he/she is lost, asks system - system gets user location and verifies if user is off the route, provides recovery instructions if so.

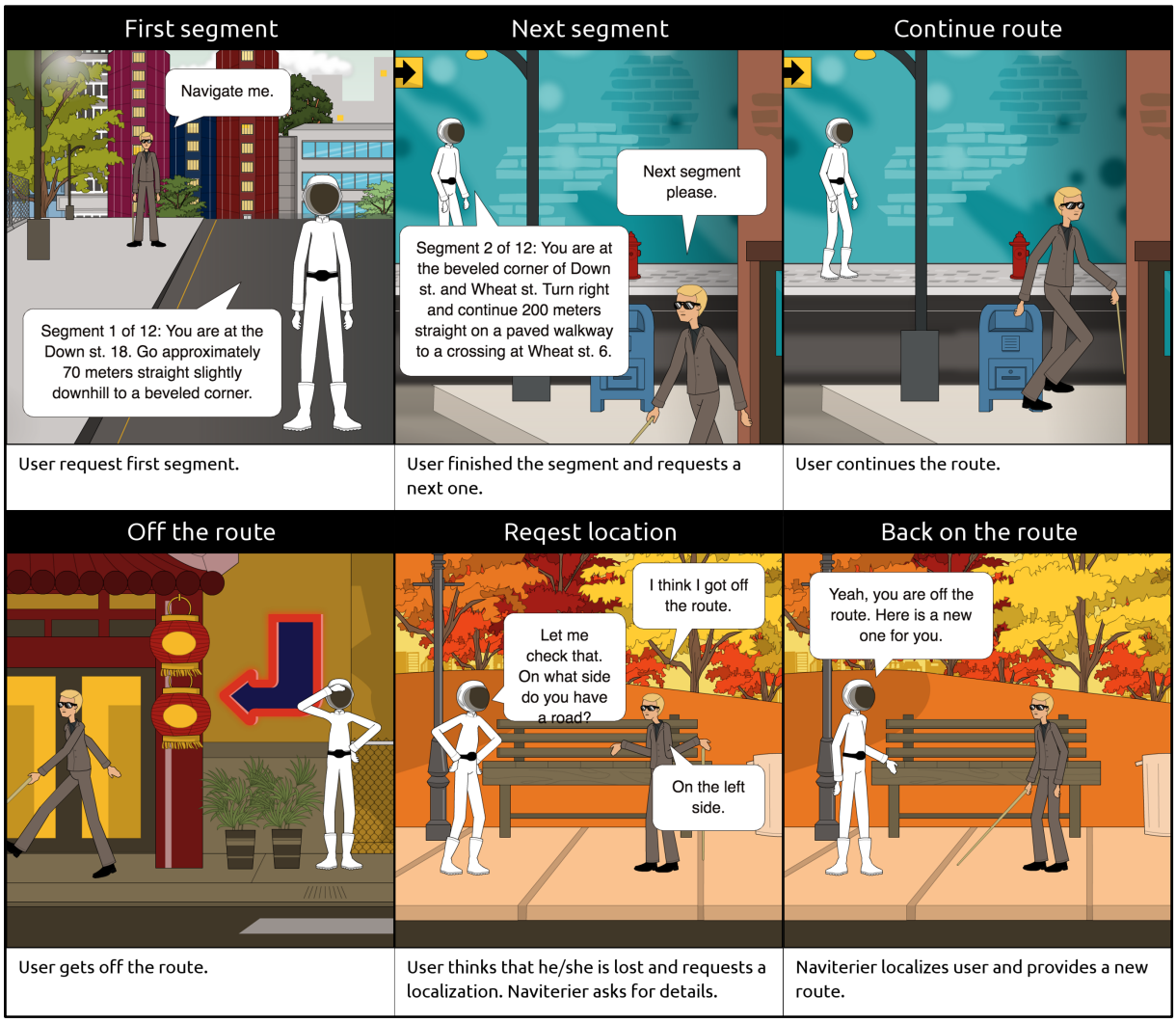


Figure 3.3: Storyboard – Navigation on the route

3.3.3 Crowdsourcing

- System tells user specifically he/she will be asked about something in the end of this segment
- User walks through the segment and asks for a new one
- System asks a question
- User answers or skips and gets back to navigation

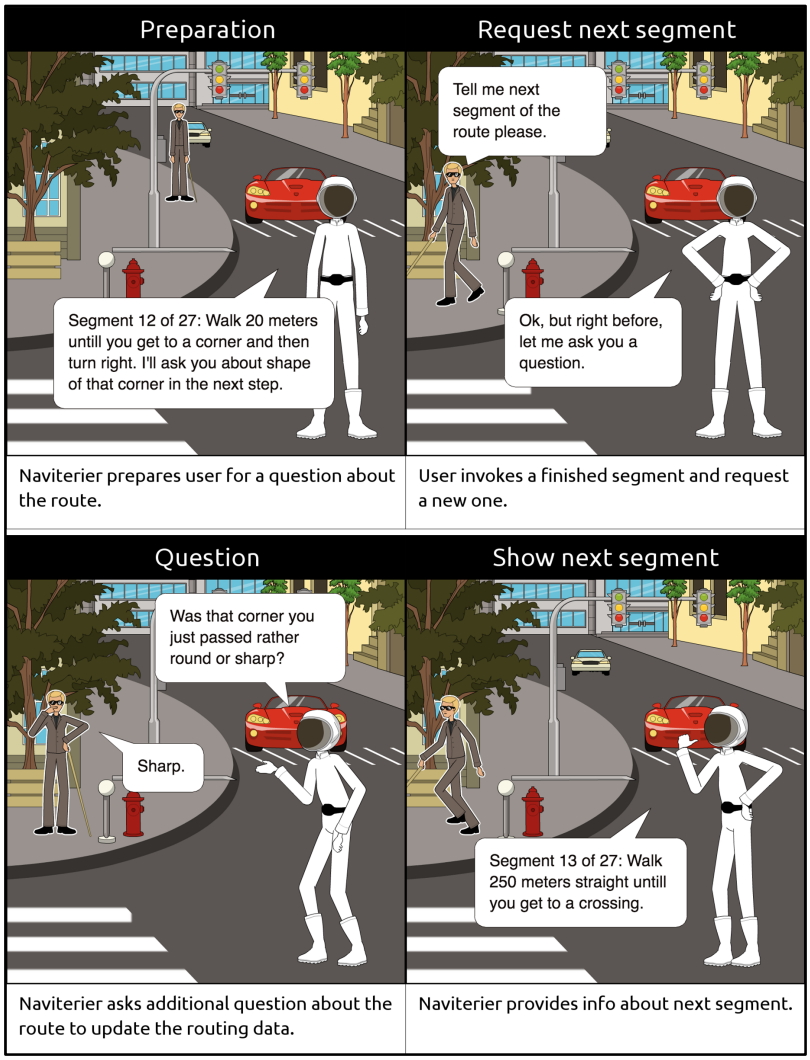


Figure 3.4: Storyboard - Crowdsourcing

3.3.4 Problem reporting

User can encounter many different problems on the route, so the reporting system will provide an option to categorize and clarify the occurred problem.

Route is impassable

- User finds out that route is impassable (something is blocking way, user can't pass)
- User reports problem as "route blocked"
- System provides alternative route

Route is inappropriate

- User finds out that route is inappropriate (bad terrain, narrow passage)
- User reports problem as "route inappropriate"
- System asks if alternate route is needed - user answers YES/NO - system responses to the request

Route information is incorrect

- User finds out that route information is incorrect (wrong terrain, slope, corner or crossing description)
- User reports problem as "incorrect information"
- System saves the report and thanks the user

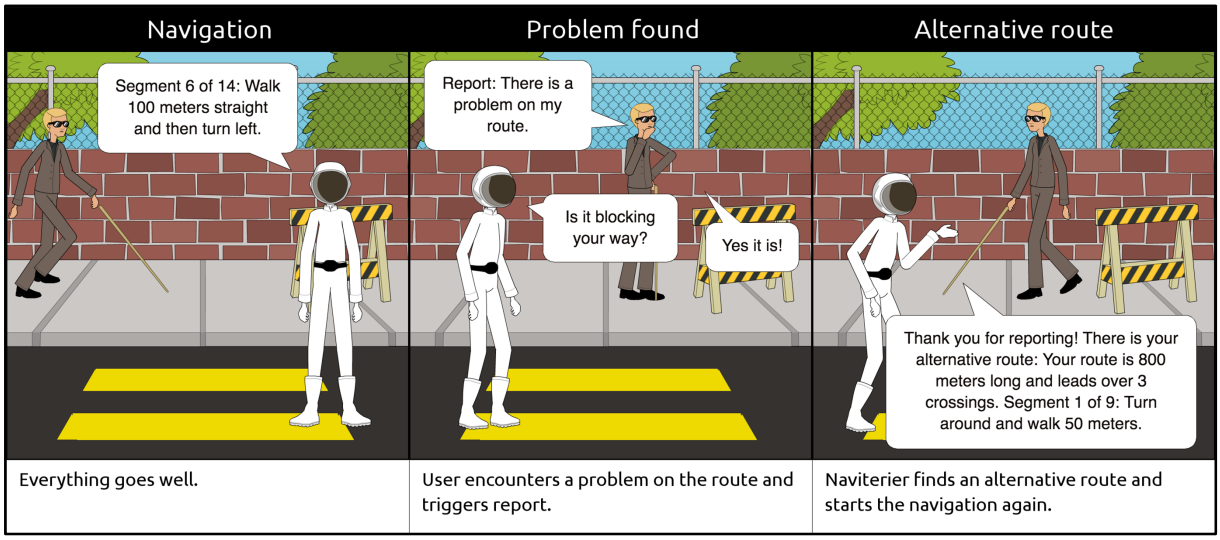


Figure 3.5: Storyboard – Problem reporting

3.4 Mockups

Based on the original Naviterier Routeplanner, use cases and storyboards were created mockups of all application screens. All mockups were created using Balsamiq Mockups software [28].

3.4.1 Homepage, address search

The user fills in destination address (or point of interest). If multiple adresses are found, user is asked to choose one.

Figure 3.6: Mockup -Homepage

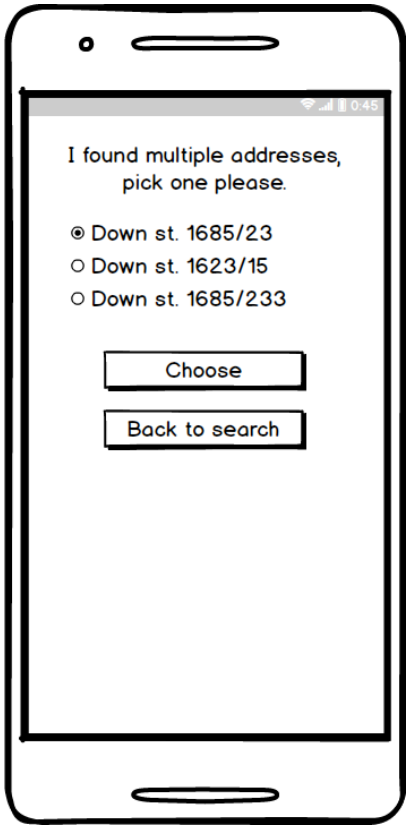


Figure 3.7: Mockup - Multiple addresses

[28] balsamiq.com/products/mockups/

3.4.2 Localization and route found

User is asked if using GPS ir required or the user knows the current location, so it can be filled in manually. User is then asked to walk a few meters in order to increase precision. In the case of GPS precision staying low even after a short walk, user is notified about the problem of bad signal and asked to fill in the address manually or walk to next street. Finally is the user asked on which side is the road to decide on which side of the street the user is.

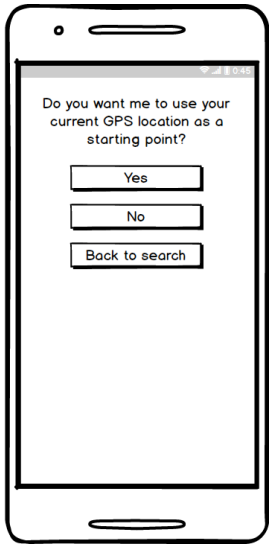


Figure 3.8: Mockup - Use GPS

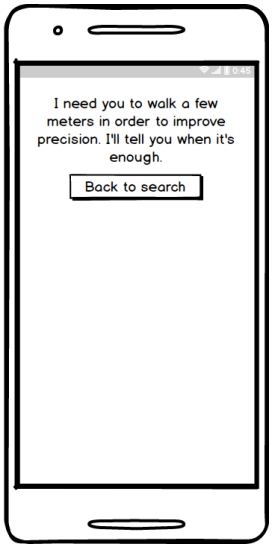


Figure 3.9: Mockup - Walk

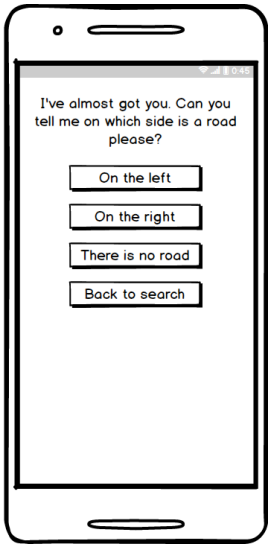


Figure 3.10: Mockup - Roadside

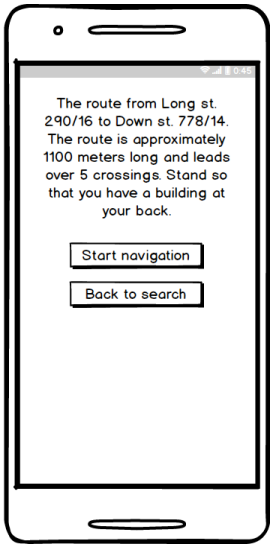


Figure 3.11: Mockup - Route found

3.4.3 Navigation and crowdsourcing

User can switch between segments of the route after successfully finishing the previous one. If crowdsourcing data is required, the user is being told at the beginning of the next segment that he/she will be asked a question about slope, terrain, roundness of the corner etc. If user feels like being lost, there is an option to request localization and check if the user is on the correct route and segment.

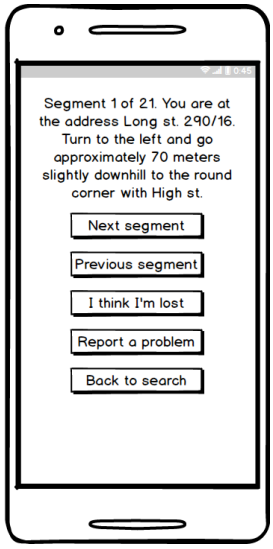


Figure 3.12: Mockup - Segment

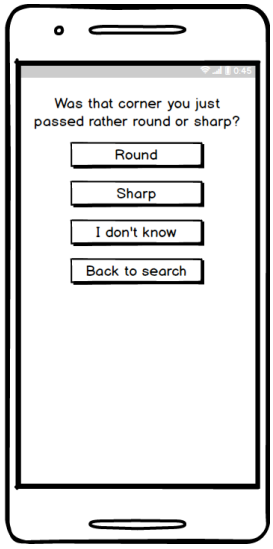


Figure 3.13: Mockup - Crowdsourcing

3.4.4 Problem reporting

When the user encounters any problem on the route, it can be reported by clicking on the „Report a problem“ button. User is then asked if there is something blocking the path. If so, problem is reported with an optional description and route is then redirected. If not, user fills in details about the problem and continues.

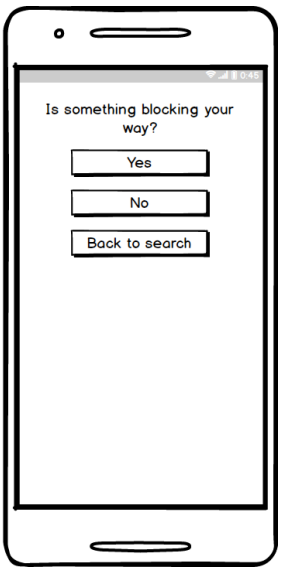


Figure 3.14:
Mockup - Report problem

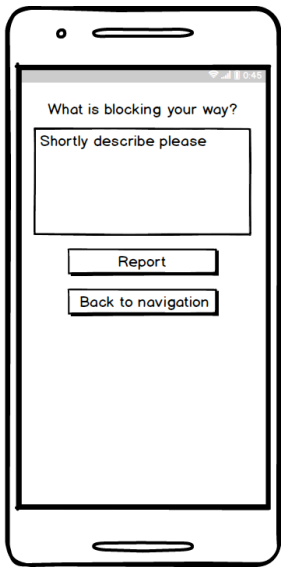


Figure 3.15:
Mockup - Blocking problem

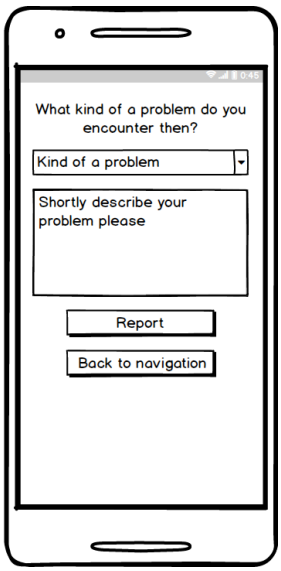


Figure 3.16:
Mockup - Non-blocking problm

3.4.5 Evaluation of mockups

All mockups were consulted with supervisor.

3.5 Low-fidelity prototypes

Low-fidelity prototypes were created after consultation of mockups with my supervisor. These mockups are plain **HTML files** (see appendix 8.4). All prototypes implement WCAG 2.0 [7] and WAI-ARIA [4] rules. Low-fidelity prototypes try to simulate the exact user interface with all accessibility features like in the final application. These are the first prototypes that can be finally evaluated or consulted with a blind user.

Some of the HTML elements like main or anchors are being omitted after consultation with blind expert due to screen readers reading unecessary information aloud and confusing the user. Most of these omitted elements are necessary to keep a website accessible and meaningful, but for an application don’t fit in the environment. All anchors have been replaced with inputs of type button or submit to keep consistency throughout the application. Most of the radio inputs were replaced by buttons for easier manipulation.

3.5.1 Evaluation of low-fidelity prototypes

Low fidelity prototypes were consulted with one sighted and one blind expert over period of one 2-hour session on iOS 10.1.1. Remarks from the evaulation are considered in making of the high-fidelity prototype.

4 High-fidelity prototype

High-fidelity prototype is a ReactJS interactive application with hardwired texts and links. It simulates functionality and behaviour of the final application. All notes and comments from previous steps were considered while creating the high-fidelity prototype.

4.1 Evaluation of the high-fidelity prototype

High-fidelity prototype was consulted with a blind expert during two 2-hour sessions. Prototypes were examined in detail on iOS 10.2, Android 6.0 and MacOS 10.12.1. Prototype was launched in web browsers, fullscreen mode and also as standalone hybrid application. Remarks from the consulting sessions were implemented in the final application.

4.2 Heuristic evaluation and cognitive walkthrough

I've conducted cognitive walkthrough and heuristic evaluation. Results showed a few minor considerable problems and also a large one regarding navigation between application screens.

4.2.1 Use cases

1. Find route from user location to destination
2. Navigate through the route
    a. Next segment
    b. Previous segment
    c. Crowd sourcing
    d. Arrival to destination
3. Report problem on the route
4. Report block on the route
5. User is lost

4.2.2 High-fidelity prototype screens

There are application screen screenshots that test results point at. All application screens are displayed below.

Figure 4.1: Prototype - Homepage

Figure 4.2: Prototype – Multiple addresses found

Figure 4.3: Prototype – On which side is the road

Figure 4.4: Prototype – Route found

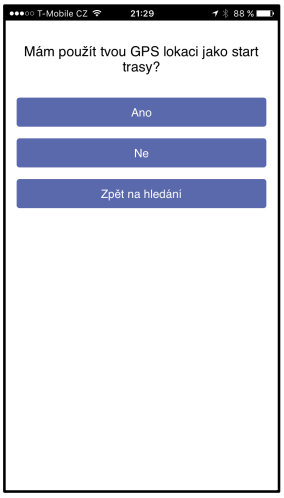


Figure 4.5: Prototype – Use GPS location as start

Figure 4.6: Prototype – Walk to make GPS more precise

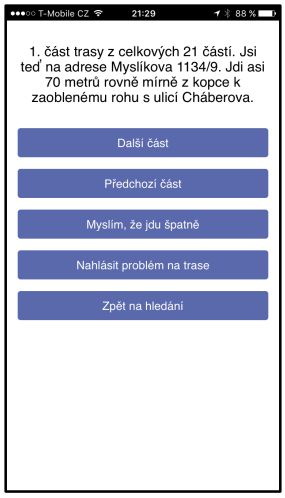


Figure 4.7: Prototype – Segment

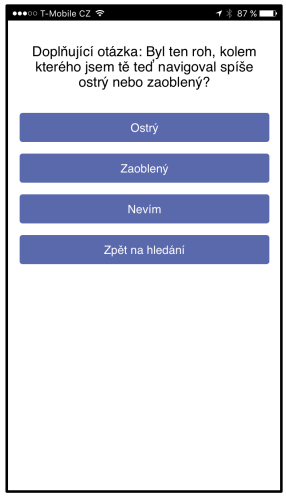


Figure 4.8: Prototype – Crowdsourcing

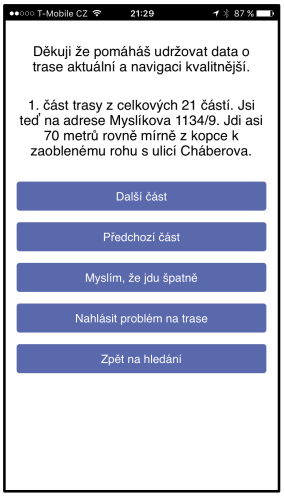


Figure 4.9: Prototype – After submitting crowdsourcing or reporting problem

Figure 4.10: Prototype – Problem reporting initiated

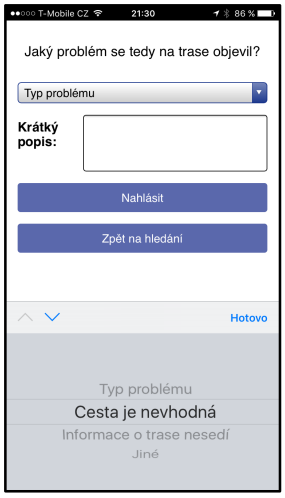


Figure 4.11: Prototype – Problem reporting, other problem

Figure 4.12: Prototype – Check location

4.2.3 Evaluation methodology

All use cases were tested using cognitive walkthrough and heuristic evaluation. Each use case was separated into atomic steps, each step is then subject to a test. Results from the test are then displayed in corresponding table where Type is either „HE“ which stands for Heuristic evaluation or „CW“ which stands for Cognitive walkthrough. Each finding is then identified to the correct screen. Additional description and suggestion belongs to every finding.

4.2.4 Cognitive walkthrough methodology

Evaluator asks himself/herself 4 questions on every atomic step. If there is unclear answer to any of these questions, it is considered as finding. I used questions mentioned in a book *Usability inspection methods* [10]. The questions are as follows:

- **Q0** - Will the user try to achieve the effect that the subtask has?
- **Q1** - Will the user notice that the correct action is available?
- **Q2** - Will the user understand that the wanted subtask can be achieved by the action?
- **Q3** - Does the user get appropriate feedback?

4.2.5 Heuristic evaluation methodology

Evaluator goes through all heuristics on each atomic step and asks himself/herself if any of them is violated. If so, evaluator marks the step and violated heuristic into results. I used 10 heuristic rules by Jaacob Nielsen [10]. The heuristics are:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

4.2.6 Results

There was a major problem found across multiple screens. Figures: **15, 16, 17, 18** encounter HE: 3. User control and freedom - **no back button**, just exit button. Suggestion: add back button.

Other minor findings are displayed respectively in tables.

Find route from user location to destination

Type	Finding	Description	Suggestion
HE	Figure 4.1. 10. Help and documentation	User doesn't know what type of input it can provide (address, description of the place, format).	Add more specific label or description.

Table 4-1: Heuristic evaluation - Fill in destination

Type	Finding	Description	Suggestion
Figure 4.2. No findings.			

Table 4-2: Heuristic evaluation – Multiple addresses found

Type	Finding	Description	Suggestion
Figure 4.3. No findings.			

Table 4-3: Heuristic evaluation – Use GPS

Type	Finding	Description	Suggestion
Figure 4.4. No findings.			

Table 4-4: Heuristic evaluation – Walk for a while

Type	Finding	Description	Suggestion
Figure 4.5. No findings.			

Table 4-5: Heuristic evaluation – On which side is a road

Type	Finding	Description	Suggestion
Figure 4.6. No findings.			

Table 4-6: Heuristic evaluation – Route found

Navigate through the route

Type	Finding	Description	Suggestion
Figure 4.6. No findings.			

Table 4-7: Heuristic evaluation – Start navigation

Type	Finding	Description	Suggestion
HE	Figure 4.7. 10. Help and documentation	User doesn't know how far is it to the destination. (Probably UI unrelated)	Add ask for distance button or embed this information into segments.

Table 4-8: Heuristic evaluation – Next segment

Type	Finding	Description	Suggestion
Figure 4.7. No findings.			

Table 4-9: Heuristic evaluation – Previous segment

Type	Finding	Description	Suggestion
Figure 4.8 & 4.9. No findings.			

Table 4-10: Heuristic evaluation - Crowdsourcing

Report problem on the route

Type	Finding	Description	Suggestion
Figure 4.7. No findings.			

Table 4-11: Heuristic evaluation – Report problem

Type	Finding	Description	Suggestion
Figure 4.10. No findings.			

Table 4-12: Heuristic evaluation – Decide problem type

Type	Finding	Description	Suggestion
HE	Figure 4.11. 7. Flexibility and efficiency of use	User is asked for a short description. Submiting a text on the go can be very hard due to speech recognition problems in noisy environment. Could lead to possibly dangerous situation while filling in the form in problematic or dangerous environment.	Description should be sctrictly marked as optional.

Table 4-13: Heuristic evaluation – Submit a problem

Report block on the route

Type	Finding	Description	Suggestion
CW	Figure 4.7. Q2	User strikes a block on the route. It may be an unexpected obstacle or user just got lost. User may click on „I think I'm lost" or „Report a problem" buttons.	Rename the „I think I'm lost" button to „Ask for location". This ensures user always clicks on „Report a problem" instead of „I think I'm lost".

Table 4-14: Heuristic evaluation – Report a problem

Type	Finding	Description	Suggestion
Figure 4.11. No findings.			

Table 4-15: Heuristic evaluation – Submit a problem

User is lost

Type	Finding	Description	Suggestion
CW	Figure 4.12. Q3	User may just encountered block on the route but application tells the user he/she is on the right route. (Not UI related)	Add repeated check for location.

Table 4-16: Heuristic evaluation – Take an action to emerge

4.3 Expert evaluation

Expert evaluation was done by a blind expert of accessibility technologies. He marked each finding with priority on the scale of 1 to 5, of which 1 is the most crucial and 5 is the least.

4.3.1 All screens

1. *[Priority: 1]* Add back button for return to previous step.
2. *[Priority: 1]* Although elements <label> are skipped while swiping through the screen reader on iOS, they cause **double reading** on Android devices. As this is tolerable behaviour on websites, it brings unecessary confusion to the application. The best option would be to hide <label> elements for screen readers and let them read only the <input> element of which are those <label> elements bound to. This can be achieved by setting aria-labelledby attribute to the <input> element. For radio and checkbox <input> elements would be the best to extend them to the whole width of the container, so user can find them with a blind cursor hovering. This is unfortunately unachievable in most web browsers, so either blind cursor hovering or double reading must be left out on these elements.
3. *[Priority: 3]* Increase height of the buttons to make them easier to find by blind cursor hovering.
4. *[Priority: 2]* Set the title of the page dynamically, because various screen readers read aloud the page title when page is focused.
5. *[Priority: 5]* Implement „back swipe“ gesture to navigate through the application faster and more freely. This uses the default browser or browser wrapper history to navigate throughout the navigation.
6. *[Priority: 5]* Attribute aria-role=“form“ is invalid. Use attribut role=“form“ instead or don’t use the role at all, because <form> element is set to the role=“form“ by default.

4.3.2 Homepage, destination search

1. *[Priority: 2]* Element <input> for destination search should have an attribute type=“search“ instead of type=“text“ so the screen reader will read aloud as searchbox instead of textbox. In addition, the submit button on virtual keyboard will be called „Search“ instead of „Submit“.

2. *[Priority: 2]* Rather remove attribute aria-hidden="false" from heading because value false is currently not fully supported in all browsers. That can lead to undefined behavior especially on Android devices where browsers implement diverse accessibility standards as warned by WAI-ARIA [4] specification. Visually-impaired users are primary, so hiding the heading for sighted users is redundant.
3. *[Priority: 4]* Attribute tabindex shouldn't be needed to use on any element except for the headings to enable focus change. Value -1 is recommended because it only allows the element to be focusable by code or screen reader but doesn't affect order of element reading or keyboard focusing.
4. *[Priority: 5]* Attribute aria-role="search" is invalid. Attribute role="search" should be used instead. The search role on <form> element is not recommended anyways as this causes screen reader to read out „Search" word on the last element inside the <form> element. That is confusing and undesirable.

4.3.3 Report a block on the route

1. *[Priority: 2]* Text field for description should be described as optional. As an alternative, the page could offer enumeration of the most common problems to choose from with an option to select „other" and then display the text field.

4.4 Conclusion

All critical findings from cognitive walktrough, heuristic evaluation and expert evaluation were fixed for the final application. In addition comboboxes were replaced by radio inputs as those are easier to manipulate for visually-impaired people if number of options is kept very low. This was suggested by a blind expert.

5 Implementation

As mentioned before, application was implemented as a hybrid application with an HTML and **JavaScript** codebase. **ReactJS** was used as library of choice for view layer due to its performance and also because of quite large amount of my experience with this library. **Redux**[29] was used as a state management tool for the application with no particular reason except for ease of use and immutability.

All the code is written in **ECMAScript 2015**[30] syntax specification with additional **JSX**[31] syntactic enhancement. The code is transpiled with **babel**[32] into standard JavaScript and HTML.

JavaScript web application is then wrapped in native code allowing it tu be installed and runned as standalone native alike application thanks to cordova. **Cordova** has quite large fan base so there is a lot of plugins, modules and modulations available on github. I used one handy geolocation plugin[33] for enabling GPS geolocalization to run in background when application is suspended.

The application is using data background of web application Naviterier as a REST API. Specifically it is being used for localization (more on that later), creation of route itinerary and search of addresses and points of interest with typing errors in mind. The method used is not standard XMLHttpRequest[34], but JSONP.[35] The application needs internet access to work.

All applications screens are available as appendix 8.1.

5.1 Crowdsourcing

Although crowdsourcing and problem reporting is implemented in the application, it is not currently fully supported in Naviterier API. The data is being logged on the server with need of human response.

5.2 Route data representation and geolocation check

The route that is user currently going is saved the whole time in the application in a form of itinerary that is being displayed to user as well as an array of geolocation points. This sorted array contains start and end points of all segments as well as all important turns or obstacles like crossings. One could imagine this array as a polyline that can display the route when drawn onto a map (Figure 5.1).

[29] redux.js.org
[30] ecma-international.org/ecma-262/6.0/
[31] facebook.github.io/react/docs/jsx-in-depth.html
[32] babeljs.io
[33] github.com/mauron85/cordova-plugin-background-geolocation
[34] www.w3.org/TR/XMLHttpRequest/ and xhr.spec.whatwg.org
[35] www.w3.org/TR/cors/#refsJSONP

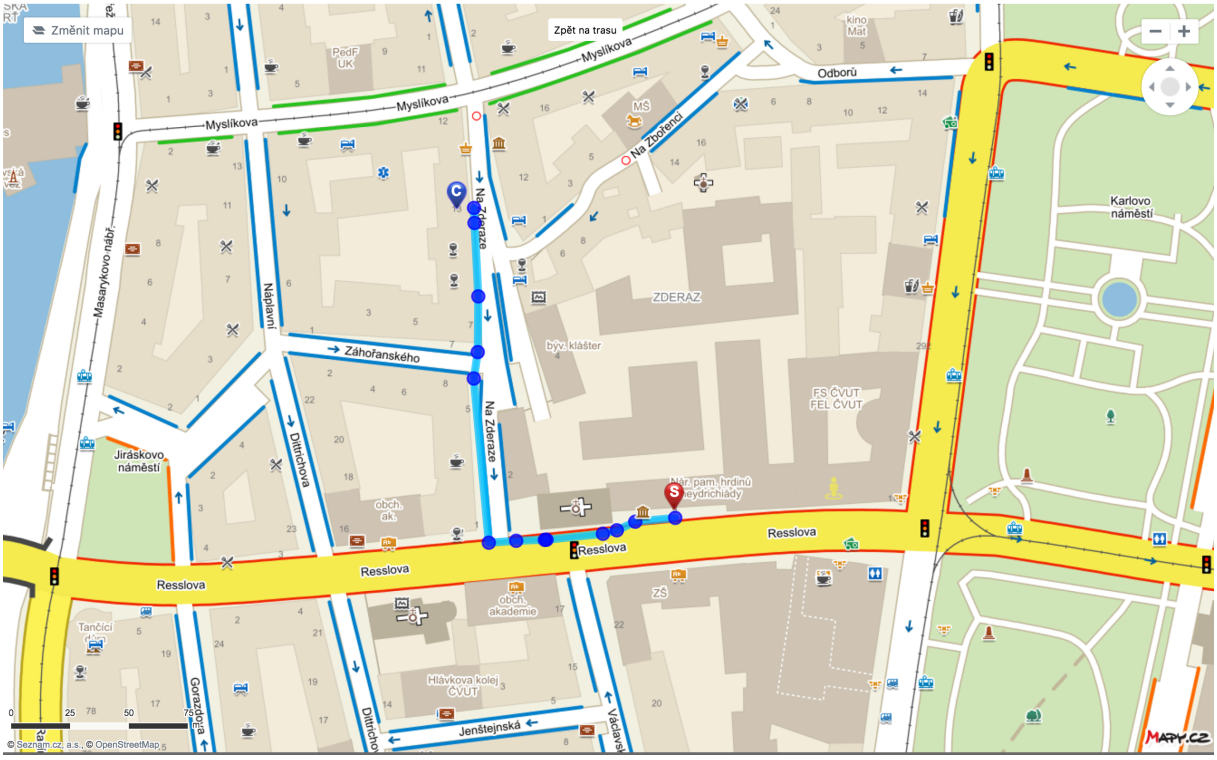


Figure 5.1: A map with marked points (blue) of route in a form of polyline (light blue)

Users can check their location if they feel lost or unsure that they are going right. GPS geolocalization is started right away and user is told to wait a second untill we can localize him. Application then uses received geolocation in form of latitude, longitude and estimated radius of accuracy to determine distance from the route. At first distance of point (user) to polyline (route) is counted as a geo distance in meters. Application uses the distance and accuracy radius to determine if user is on or very close to the route by subtracting the radius from the distance. Application responses to user by saying he/she is off the route if he/she is 25 or more meters away from the route. User is told it's currently impossible to localize him/her if 30 seconds past without a response from GPS.

5.3 GPS localization

There are two options of determining start location when searching for a route. User can either input his/her start location manually or use GPS localization. In the case of GPS localization is user asked to walk about 30 meters in a direction from a corner of the street to its middle. Application captures all received geolocation points in a form of object containing latitude, longitude and accuraccy radius. A mean point is created if enough points were captured and distance walked. User is then asked on which side was road. If no road exists, user is told that we can't localize him accurately on pavements without roads nearby at the moment. The mean point is send to Naviterier API along with answered road side and last captured point (current location) to determine current address.

The mean point calculation is very important, because in calculation with current position determines a tangent vector of movement and thus direction the user is moving and facing to.

The capturing algorithm in a form of pseudocode is as follows.

```
geopoint captured DO
   IF   point radius < 24 meters OR time elapsed > 35 seconds
   AND  number of saved points >= 8
   AND  distance between first captured point and latest point >= 30 meters
   THEN
      calculate mean point
      get address
   ELSE
      save point
END geopoint captured
```

And here is mean point calculation. Each property of a point object (latitude and longitude) is calculated independently. $\mathbf{M_l}$ is latitude or longitude of the mean point. $\mathbf{C_p}$ is count of captured points. $\mathbf{p_i}$ is a point latitude or longitude. $\mathbf{a_i}$ is point accuracy radius in meters. This formula ensures that inaccurate captured points have lesser impact on the calculated mean point than more accurate points.

$$M_l = \sum_{i=0}^{C_p} \frac{p_i}{a_i} \times \frac{1}{\sum_{i=0}^{C_p} \frac{1}{a_i}}$$

Figure 5.2: Mean point formula

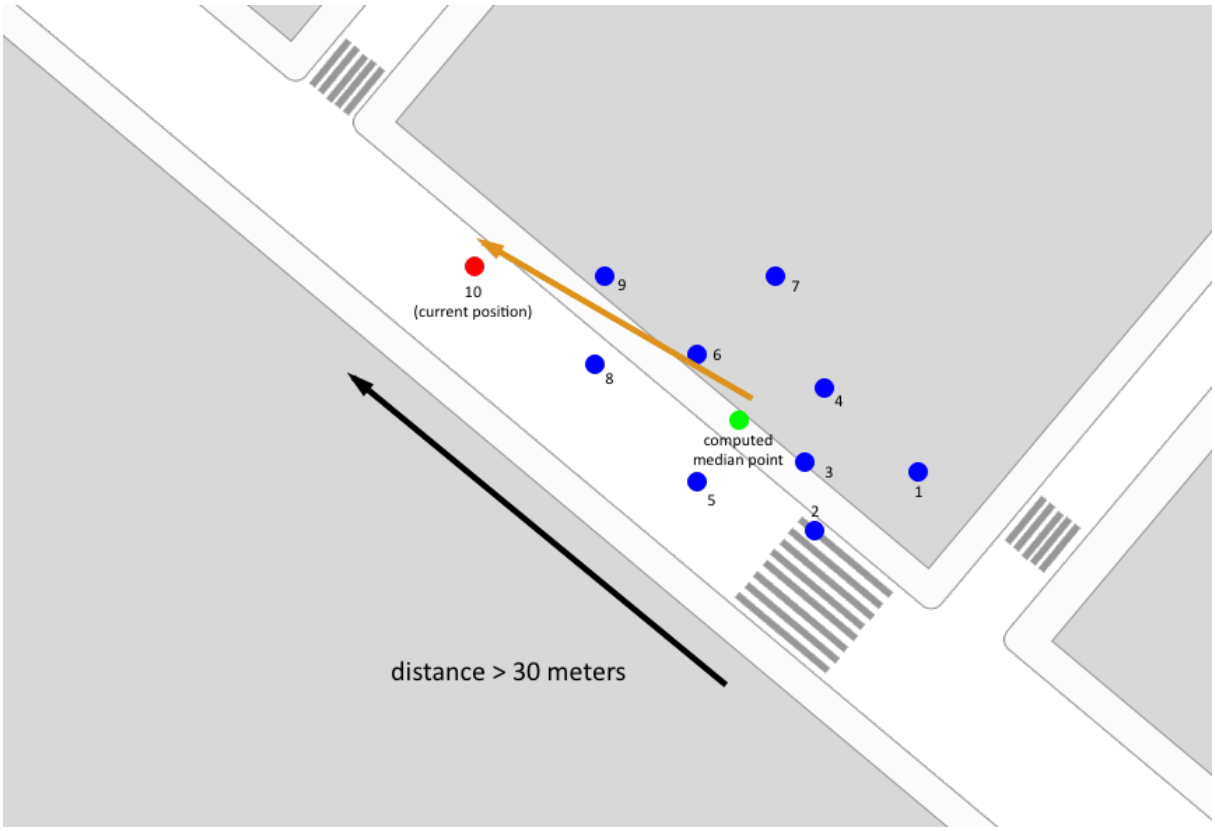


Figure 5.3: GPS localization

5.4 Screen reader focus management

For management of screen reader focus I used a combination of focus() function that is available as a property of any Document Object Model node. As well as role="alert" attribute. iOS doesn't allow calling focus() unless it's a synchrounous direct response to user action. That is partly a problem because sometimes I need to respond with managing screen reader focus after application induced action like receiving error or GPS position. I had to invent a clever way how to do that. And in the end it's simpler than I thought it would need to be. Anytime user takes an action like clicking a button application injects a recursive function that waits for anything to gain focus on in the action callback. This way is the function put into a stack of direct response methods in the virtual machine waiting there untill I allow it to focus onto something when I need to. Because it sits in the stack of synchronous direct responses iOS allows a call of focus() even a long time after action takes place. Unfortunately it can be done only once for every action. That is not a problem anyway because alert="role" and native notifications were used for any other necessary announcement of asynchronous events and changing application state.

5.5 Platform support

The application can be installed on and natively supports all modern smartphones using Android 4.4.4 and iOS 8 or newer versions as well as web browsers of these and other devices including desktops

6 Evaluation of the final application

The last step is testing of the final application. At first I did a pilot test with a blind expert. He gave me a few suggestions that I managed to get into the application or test procedure itself. My supervisor and his collegue helped me get participants for this test by sending an email to a community of visually impaired. I received response from 6 participants matching our screening. We were looking for visually-impaired people using a touchscreen smartphone with iOS or Android operating system. There was no restriction in age, gender or amount of experience with touchscreen smartphone. Optional goal was to have an equal amount of male and female, equal amount of Android and iOS users and age span on the interval from 20 to 60 years old. The application was tested with these 6 participants on the street in city of Prague.

6.1 Participants

Notes	Participants					
	P1	P2	P3	P4	P5	P6
Gender	male	male	female	female	male	male
Age	28	49	27	26	32	39
Devices used	iPhone 6	iPhone 5s	iPhone 5s	Samsung Galaxy (Android)	several Android + iOS	iPhone 5s
Screen reading technique used	swiping through next and previous element	swiping through next and previous element	swiping through next and previous element	sliding finger in a zic-zac motion from top to bottom searching elements on the screen	swiping through next and previous element	swiping through next and previous element
Touchscreen smartphone usage	daily	daily	daily	daily	daily	daily
Touchscreen smartphone ownership	1 year	1 year	2 years	3 months	7 years	2 years
Impairment onset	cogenitally	cogenitally	cogenitally	cogenitally	cogenitally	late
Impairment type	slight vision of light	blind	slight vision of light	blind	blind	blind
Guide dog	no	no	no	no	no	no

Table 6-1: Participants

We got 2 females and 4 males. Age mean value of all participants is 33.5. Age mean value of females is 26.5. And age mean value of males is 37. 4 Participants are iOS users, 1 Android

user and 1 user of both platforms. 5 participants are experienced with touchscreen smartphones. One participant is in learning phase of smartphone usage.

6.2 Apparatus

All participants had a lavalier microphone clipped to their jackets during the whole test. Lavalier microphone used was A.Lav by Aputure[36]. This lavalier microphone was connected to an audio recorder. Audio recorder used for this test was H1 by ZOOM[37]. Preparation, all instructions, the test itself and questionnaire after test are a parts of those audio records. Participants were given my iPhone 7 with pre-installed application as a testing device. One reason for that is a problem with distribution of unreleased application on iOS devices and the other is consistency and better control of the test environment. No participant had a problem with that. Participant P4 as an Android user was given time in advance to try off all accessibility functions of the phone. Because she was a touchscreen smartphone beginner, and both iOS and Android accessibility controls are very similar. She had no problem at all with the switch and even confirmed that she doesn't see any difference between this and her phone.

6.3 Procedure

Each participant was assured that we are testing the application, not participants. They were given audio recorder with lavalier microphone and asked to think aloud about anything they'll face during or after the test. I assured them that I'll be nearby the whole time, I just won't intervene the test untill it's necessary. They were given the information that they'll be asked to get somewhere with the application without any help from anyone else. The route will be maximally a few hundreds long and there may be a pedestrian crossing. They may ask for help on pedestrian crossing or any dangerous obstacle. In the case of possible injury I am going to step in. I will also try to block out any heedless tourist or pedestrian that may cause an injury to the participant. They were also told that we will spend some time on a busy street and the microphone may struggle picking up any sound, so try to talk louder than normal. In case of speech input I can provide sight support because the confirmation feedback of native speech recognition is very faint and they may don't hear it. Every participant knew from our phone calls and emails that they will be given an iPhone 7 with pre-installed application.

Before the test I gave participants short introduction of the application. In a few senteces said what is the application capable of. They were given the information that application can give you detailed navigation itinerary from point A to point B. They can search for start and destination in a form of addresses and points of interest. They can also use GPS for localization instead of manual input of current location. The application can also check their location on the route and will probably ask in advance for additional information about the route. They will be given enumeration of options to answer. I also told them that they can use the application to report any encountered problem and the application may try to give them some instruction to recover from the problem.

[36] aputure.com/en/A.lav.html
[37] zoom-na.com/products/field-video-recording/field-recording/zoom-h1-handy-recorder

After accepting all information and agreeing with the test. I walked them to address Resslova 11 and turned them facing downhill in the direction of expected route. They were given the testing smartphone and I asked them to walk me to address "Na Zděraze 13". You can see the testing route on the figure 6.1. The route is about 350 meters long. After finishing this route I asked them to take me to "some Pilsner restaurant at Karlovo náměstí". The purpose of this was to test searching points of interest. They were asked to enter their current location manually because they know it (and I also repeated that address aloud). I stopped them after walking about 20 meters. They were told that there is imaginary wall that is blocking their way. Possibly whole building, but application tells them to go ahead. What would they do in that case? After answer and taken action in the application was test ended. We continued with a short questionaire.

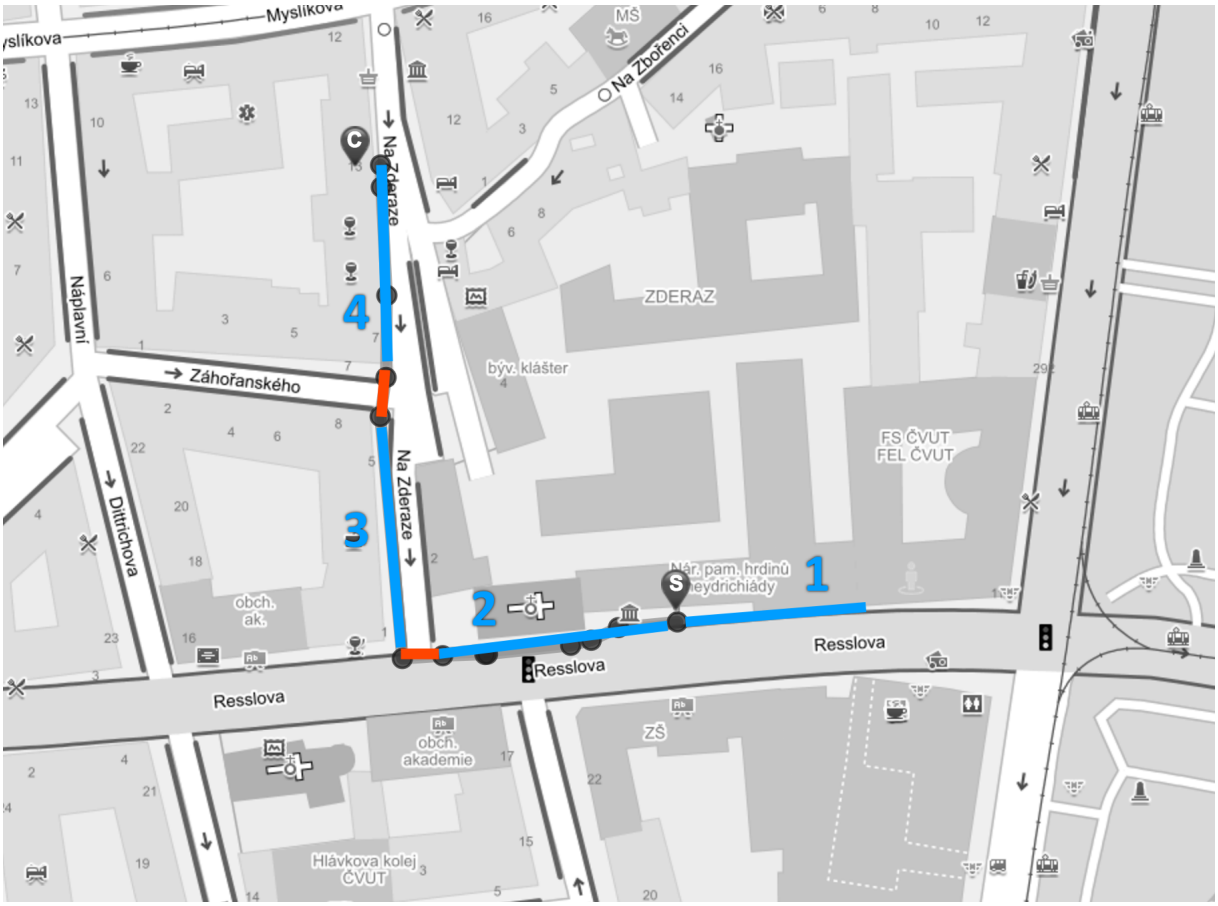


Figure 6.1: Testing route; blue segments; red pedestrian crossings

6.4 Measures

The application logged to our server every user action (button press, screen change, inputs), the feedback they received and also any information that application received in the background (responses from REST API, captured geopoints and other). This is support data to the audio records.

The route is separated into segments as seen on figure 6.1. Each segment consists of some simple task.

Segment 1 is a GPS localization. Participants are supposed to walk straight and get localized exactly on address "Resslova 9" where their route starts.

Segment 2 is a first segment on the route. Participants should go straight to the pedestrian crossing. But as seen in appendix 8.3-A, 8.3-B and 8.3-C there is a lot of irregularities of the building and pavement. There is also another pedestrian crossing sideways.

Between segment 2 and 3 is relatively safe pedestrian crossing with low traffic in one direction. See appendix 8.3-C.

Segment 3 (appendix 8.3-D) announces that participant will be asked about slope of the pavement at the end of the segment. There is also a small building site that may be an obstacle as seen on apendix 8.3-E.

Between segment 3 and 4 is very wide and round pedestrian crossing with no traffic at all. See appendix 8.3-F.

Segment 4 (appendix 8.3-H) stars with a small obstacle (parked bicycles) as seen on appendix 8.3-G. The end of the route is this building entrance (appendix 8.3-I). Street continues about 40 meters to an intersection. See appendix 8.3-J.

I wrote down how well did each participant on each segment and crossing. What problems they had and what they did very well or better than expected.

I prepared a short after-test questionnaire. Answers can be open, a value, or based on Likert scale [11].

ID	Question
Q1	How do you like the application?
Q2	Would you go to an unknown place just with this application alone?
Q3	Would you use the application to search for exact addresses or points of interest?
Q4	What did you like about the application?
Q5	What did you dislike about the application?
Q6	If you could change anything in the application, what would it be?
Q7	For how long routes would you use the application?
Q8	How often would you use the application?
Q9	How many crowdsourcing questions would you be able to answer on a route of 10 segments without it being disturbing?
Q10	Did it hinder you to go 30 meters straight on an unknown street in order to localize you?
Q11	How many meters would you be willing to go on an unknown street for the GPS localization to determine your exact location?
Q12	Is there anything that could ease that process for you and make you go further in order to determine your exact location?

Q13	If you'd encounter a problem on the route, would you report it?
Q14	What kind of a problem would you most likely report?
Q15	Were the data from the application correct?
Q16	What data were incorrect?
Q17	Were the instructions from the application clear and easy to follow?
Q18	What instructions were not clear or easy to follow?
Q19	Was the application comfortable to use on noisy and busy street?

Table 6-2: Table of asked questions

ID	Value answer	Open answer	Likert scale answers and values				
			0	1	2	3	4
Q1			dislike very much	rather dislike	neither	rather like	like very much
Q2			definitely no	rather no	neither	rather yes	definitely yes
Q3			only addresses	rather addresses	both the same	rather POI	only POI
Q4		x					
Q5		x					
Q6		x					
Q7	x						
Q8	x						
Q9	x						
Q10			definitely did not	rather did not	neither	rather did	definitely did
Q11	x						
Q12		x					
Q13			definitely would not	rather would not	neither	rather would	definitely would
Q14		x					
Q15			none	most not	some	most	all
Q16		x					
Q17			none	most not	some	most	all
Q18		x					
Q19			definitely was not	rather was not	neither	rather was	definitely was

Table 6-3: Table of possible answers

6.5 Results

I collected results from the audio records, application logs and a questionaire. See appendix 8.2.1 for notes how well was each segment finished. Questionnaire answers can be seen in appendix 8.2.2.

Participants finished the route in these times.

P1	P2	P3	P4	P5	P6
14 minutes	21 minutes	25 minutes	14 minutes	9 minutes	17 minutes

Table 6-4: Times needed to finish the route

Mean time needed to finish the route was 16.6 minutes, mean time to finish of people who did not encountered any larger problem was 13.5 minutes.

All participants successfully finished all given tasks eventually. Except for arriving exactly to the destination. But more on that later. 2 participants (P2 and P3) experienced a loss on the route. You can see details on that in appendix 8.2.

P2 incorrectly assumed the application will stop him automatically when gets off the route, did not read the instructions neither switched segments assuming this happens automatically. Got to the "check location" screen and walked headlessly untill he bumped to a dead-end when already off the route. He assumed that application check the location in the background. Then he got back and continued without any problems.

P3 got confused by irregular building wall in segment 2 and stopped too early assuming it is a corner. Didn't reach the crossing (while standing about 5 meters away from it) after several minutes eventually recovered by finding the crossing. Continued without any further problems. Badly assumed when nearing the destination that the application will tell when to stop. Thus passed the destination by more than 40 meters.

Probably the largest problem on the route was the second segment where stands a historical building with a lot of features carving into the pavement. Participants were confused thinking they walked to a corner of the street.

Half of the participants passed the destination because they expected a distance announcement or that the application will stop them when they arrive to the destination. None of the participants was sure if arrived to the destination. 3 of the 6 participants arrived exactly or with maximal 3 meters error. The other half passed the destination by atleast 20 meters. These results may do not correspond with real life results as participants walked to an unknown unmarked door. They would usually walk to a known place or had someone to pick them up. But that doesn't delete the fact that it's a large problem that needs to be fixed.

Participants tend to use GPS localization in the third part of the test even though they had the instruction to insert it manually because they know it. This probably means they liked the feature that they'd use it even at known places just for the convenience.

7 Conclusion

I managed to create fully functioning multi-platform navigation application for visually-impaired people with all required functionality. Users can search for points of interest or addresses with typing errors independence. Check if they went off the route. Report a problem and use GPS to localize themself. Users are also being asked occasionally about additional data on the route. The application runs on Android 4.4.4 and newer as well as on iOS 8 and newer. No Symbian devices currently supported. Support for selected requirements matching Symbian devices is possible.

The design was done in iterations and each iteration was consulted with blind or sighted expert. Evaluation of the final application proved that it is able to successfully fullfill its purpose. All participants reported they liked the user interface and simplicity of the application. They appreciated detailed information and instruction.

Testing also revealed some problems that need to be fixed. Most of the problems may be just minor, but there are also some crucial. The most crucial problem is lack of distance check so users don't know how far is their destination and maybe even the end of current segment. There were also some minor hickups with accessibility feedback that need to be fixed. Participants also reported that they would appreciate if the application talked to them as they walked their 30 meters to localize them. They weren't sure if they already finished the distance or not.

2 participants encountered major problems on the route. Both of them were caused because of lack of distance check and bad assumption of non-existing functionality. The application should clearly state that it does not respond automatically to user walk. Only to user actions expressed by buttons.

7.1 Future work

Work on this navigation application continues. The next thing on the list is adding distance checker right after fixing minor hickups with feedback. There will be also probably added more talkative feedback for the localization phase. I am also playing with an idea of adding location and distance check in the background with an interval of about 10, seconds that wouldn't drain battery but kept users more securely localized. The application could even announce remaining distance using this interval. An integration with headset buttons could also help lower the distraction and free up the smartphone-holding hand. There is currently a plan of including user accounts and customization including favorite locations and routes as well as last used route or location. Some settings like desired battery drain level (and thus the interval of checks) should be customizable too.

References

[1] L. Richardson, M. Amundsen and S. Ruby, RESTful Web APIs, O'Reilly Media, 2013.

[2] J. Thorp and S. H. Lawton, “Notes on User Centered Design Process (UCD),” Web Accessibility Initiative (WAI), World Wide Web Consortium (W3C), 31 March 2004. [Online]. Available: https://www.w3.org/WAI/redesign/ucd. [Accessed 12 December 2016].

[3] J. Balata and Z. Mikovec, “Why visually impaired people resist to adopt touchscreen smartphones in the Czech Republic?,” *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom),* pp. 000049-000054, 16 October 2016.

[4] J. Craig, M. Cooper, L. Pappas, R. Schwerdtfeger and L. Seeman, “Accessible Rich Internet Applications (WAI-ARIA) 1.0,” Web Accessibility Initiative (WAI), World Wide Web Consortium (W3C), 20 March 2014. [Online]. Available: https://www.w3.org/TR/wai-aria/. [Accessed 25 October 2016].

[5] P. J. Adam, “iOS vs. Android Accessibility,” [Online]. Available: http://pauljadam.com/iosvsandroida11y/. [Accessed 16 November 2016].

[6] W. Chisholm, G. Vanderheiden and I. Jacobs, “Web Content Accessibility Guidelines 1.0,” Web Accessibility Initiative (WAI), World Wide Web Consortium (W3C), 5 May 1999. [Online]. Available: https://www.w3.org/TR/WAI-WEBCONTENT/. [Accessed 25 October 2016].

[7] B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, W. Chisholm, J. Slatin and J. White, “Web Content Accessibility Guidelines (WCAG) 2.0,” Web Accessibility Initiative (WAI), World Wide Web Consortium (W3C), 11 December 2008. [Online]. Available: https://www.w3.org/TR/WCAG20/. [Accessed 25 October 2016].

[8] C. Power, A. Freire, H. Petrie and D. Swallow, “Guidelines are only half of the story: accessibility problems encountered by blind users on the web,” *CHI '12 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* pp. 433-442, 5 May 2012.

[9] D. Špinar, Tvoříme přístupné webové stránky, Zoner Press, 2004.

[10] J. Nielsen and R. L. Mack, Usability Inspection Methods, New York, NY: John Wiley & Sons, 1994.

[11] S. A. McLeod, “Likert Scale,” 2008. [Online]. Available: www.simplypsychology.org/likert-scale.html. [Accessed 8 January 2017].

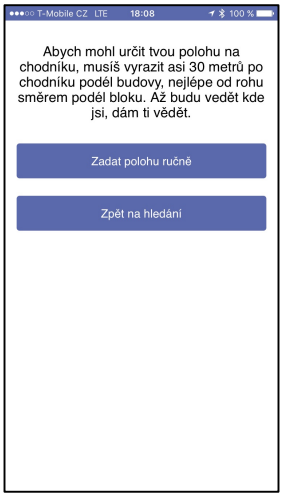
8 Appendix

8.1 Screens of the final application

Appendix 8.1-A: Homescreen

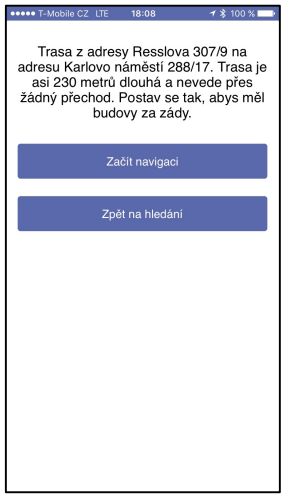
Appendix 8.1-B: Multiple addresses found

Appendix 8.1-C: Route destination found



Appendix 8.1-D: User is asked to walk at least 30 meters straight

Appendix 8.1-E: User is asked on which side is a road

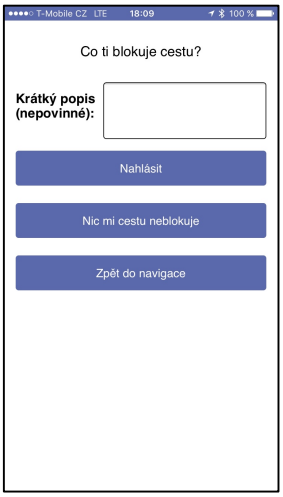


Appendix 8.1-F: Start found, route started

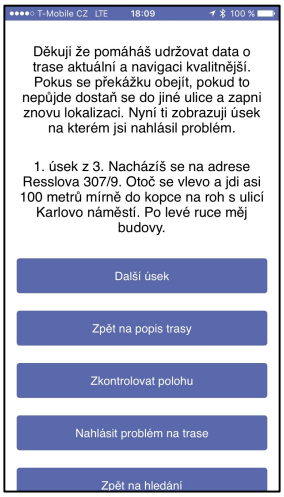
Appendix 8.1-G: Segment of navigation itinerary

Appendix 8.1-H: Location check

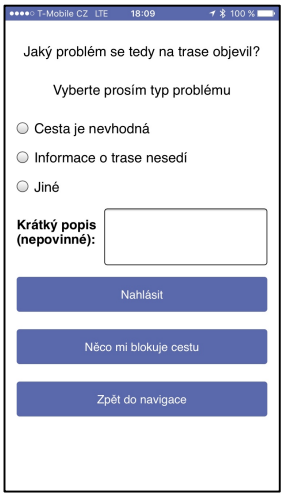
Appendix 8.1-I: Problem report initiated



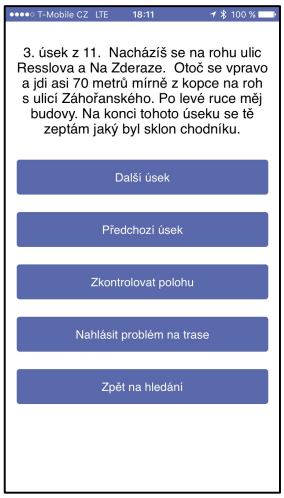
Appendix 8.1-J: Reporting blocking problem



Appendix 8.1-K: Problem reported, returned to last segment with message



Appendix 8.1-L: Report a non-blocking problem



Appendix 8.1-M: Crowdsourcing request is announced ahead

Appendix 8.1-N: Crowdsourcing question

8.2 Final application evaluation results

8.2.1 Data from the route

Appendix 8.2-A: Segment 1

P1: After walking around 20 meters, impatiently switches to entering current location manually. Later switches back to GPS and walking.
P2: Successfully localized.
P3: Successfully localized.
P4: Impatient about the application not talking. Successfully localized.
P5: Successfully localized.
P6: Successfully localized.

Appendix 8.2-B: Segment 2

P1: Confuses building irregularity with corner. Realizes he is not at the crossing yet and finishes the segment.
P2: Doesn't use "next button". Switches to "Check location" and awaits auto updates. Walks off the route, correct street and direction, just incorrect side of the road. He was corrected from the wrong assumptions he made. Successfully walks back and finishes segment. Later responded that he did not read instructions correctly.
P3: Skipped the route found screen. Get's confused by the irregular building. Not sure if at the corner or not, but "check location" reports just that she is on the route (which is true). After several minutes wandering around and walking back and forth forgets what she was looking for and gives up. I step in and ask her what can she do. She says she can either get back to start or continue walking down the street and hope she finds the corner. I asked her to do anything she thinks is the best. After walking 5 more meters down the streets finds the crossing.
P4: Bumps into irregularities but recovers and finds the crossing successfully.
P5: Successfully walks to the crossing.
P6: Finds the crossing. Get's little confused by historical building on the right.

Appendix 8.2-C: Crossing 1

P1: Crosses without assistance or hesitance.
P2: Crosses without assistance.
P3: Crosses without assistance.
P4: Crosses without assistance.
P5: Looking for a sound feedback on pedestrian crossing. No sound feedback available. Crosses without assistance.
P6: Crosses without assistance.

Appendix 8.2-D: Segment 3

P1: Reports building site as an obstacle. Successfully answers crowdsourcing.
P2: Successfully answers crowdsourcing.
P3: Successfully answers crowdsourcing.
P4: Successfully answers crowdsourcing.
P5: Bumps into building site. No report. Successfully answers crowdsourcing.
P6: Forgot to pay attention to pavement slope. Not sure of his answer.

Appendix 8.2-E: Crossing 2

P1: Crosses straight without assistance.
P2: Crosses straight without assistance.
P3: Crosses straight without assistance.
P4: Crosses straight without assistance. Bumps into a trash can.
P5: Crosses slightly off the pedestrian crossing, corrects himself.
P6: Crosses slightly off the pedestrian crossing, corrects himself.

Appendix 8.2-F: Segment 4

P1: Bumped to obstacle, no report. Not sure if arrived to destination. Arrived exactly.
P2: Bumped to obstacle, no report. Not sure where is the destination. Passed on the destination by around 20 meters.
P3: Not sure where is the destination. Awaits the application to tell her when to stop. Passed on the destination to the next corner. At first thinks about crossing to next street. Then realizes the application won't tell her and she needs to count the meters. Makes a bad guess of distance. Doesn't find the destination correctly.
P4: Bumped to an obstacle, no report. Not sure where is the destination. Passed the destination by about 10 meters. Thinks she is not correct and gets back to the destination correctly.
P5: Doesn't know where is the destination. No distance check available. Passes the destination. Checks location and walks back right to the destination.
P6: Passed destination by about 30 meters. Tries to check location but receives positive answer dues to short distance.

Appendix 8.2-G: Points of interest search

P1: Successfully searched for the right place.
P2: Successfully searched for the right place.
P3: Successfully searched for the right place.
P4: Successfully searched for the right place.
P5: Encounters a hickup with keyboard stuck open and screen reader not reading the content. Recovers later and successfully searched for the right place.
P6: Successfully searched for the right place.

Appendix 8.2-H: Unexpected wall in front

P1: Would check location, asked for assistence or try to go to another street. Report if data not correct.
P2: Would walk back and try again. Then checked location and reported eventually.
P3: Would try to recover at first by walking to another street. Then tried to localize again.
P4: Would try to get back and then reported if sure it's not her mistake.
P5: Would try to walk back and check location. Would try to go around. Reported only if really blocking the way.
P6: Would think it's his fault. Reporting only if certainly sure there is something wrong with the data.

8.2.2 Data from questionnaire

Q1: How do you like the application?					
P1	**P2**	**P3**	**P4**	**P5**	**P6**
like very much	like very much	like very much	like very much	like very much	rather like

Appendix 8.2-I: Q1

Q2: Would you go to an unknown place just with this application alone?					
P1	**P2**	**P3**	**P4**	**P5**	**P6**
definitely yes	rather yes	definitely yes	definitely yes	definitely yes	definitely yes

Appendix 8.2-J: Q2

Q3: Would you use the application to search for exact addresses or points of interest?					
P1	**P2**	**P3**	**P4**	**P5**	**P6**
rather addresses	both the same	rather addresses	rather addresses	both the same	both the same

Appendix 8.2-K: Q3

Q4: What did you like about the application?					
P1	**P2**	**P3**	**P4**	**P5**	**P6**
system of segments, detailed descriptions	simple controls, clean UI, very detailed description of segments and surroundings (like a frined would talk to me), division to simple and short segments	detailed description of route, options like check location, report a problem etc.	details, check location, GPS localization	easy controls, seamless functionality, asked questions to refine geolocation	simple interface

Appendix 8.2-L: Q4

Q5: What did you dislike about the application?					
P1	**P2**	**P3**	**P4**	**P5**	**P6**
screen reader feedback sometimes not correct, the need to walk 30 meters in order to get GPS location	missing continuos reporting of current location	Sometimes too much information at once, missing detail about building irregularity during second segment	missing distance counter, missing alert before zic-zac building wall in second segment	no continuous announcing of location, need to hold a phone in hand	no continuous distance announcement, need to hold a phone in hand

Appendix 8.2-M: Q5

Q6: If you could change anything in the application, what would it be?					
P1	P2	P3	P4	P5	P6
usage with external devices like headphones with buttons	add continuous reporting of current location or report current location on demand, show distance to the end of the segment	-	a button to check distance from destination	add continuous location check, categorized search of nearby POI, in check position	add continuous announcement of distance to end of the segment and to the destination, prepare route in advance and use then offline

Appendix 8.2-N: Q6

Q7: For how long routes would you use the application?					
P1	P2	P3	P4	P5	P6
any length	mostly 1-8 streets	any length	any length	mostly 2-8 streets	any length

Appendix 8.2-O: Q7

Q8: How often would you use the application?					
P1	P2	P3	P4	P5	P6
1-2 times a week	3-4 times a week	3-4 times a month	once a month	3 times a month	1-2 times a month

Appendix 8.2-P: Q8

Q9: How many crowdsourcing questions would you be able to answer on a route of 10 segments without it being disturbing?					
P1	P2	P3	P4	P5	P6
3-4	2-3	2	2-3	every segment	atleast 5. possibly every segment

Appendix 8.2-Q: Q9

Q10: Did it hinder you to go 30 meters straight on an unknown street in order to localize you?					
P1	P2	P3	P4	P5	P6
rather did	rather did not	rather did not	rather did	rather did not	rather did not

Appendix 8.2-R: Q10

Q11: How many meters would you be willing to go on an unknown street for the GPS localization to determine your exact location?					
P1	P2	P3	P4	P5	P6
15 if possible, 50 maximally	20 if possible, 50 maximally	30 if possible, 50 maximally	20 if possible, 60 maximally	maximally 30	maxiamlly 50

Appendix 8.2-S: Q11

Q12: Is there anything that could ease that process for you and make you go further in order to determine your exact location?					
P1	P2	P3	P4	P5	P6
rating of the route by danger, would walk maximum range on quiet street, minimum on busy one	continuous location reporting	-	application constantly speaking and giving instructions	continous announcement of refined location	say road side in advance and then go with application announcing remaining distance

Appendix 8.2-T: Q12

Q13: If you'd encounter a problem on the route, would you report it?					
P1	P2	P3	P4	P5	P6
rather would	definitely would	definitely would	rather would	rather would	rather would

Appendix 8.2-U: Q13

Q14: What kind of a problem would you most likely report?					
P1	P2	P3	P4	P5	P6
any dangeroues or path changing problem	pavement trench, any long-term blockade on the route	something long-term blocking the way	anything dangerous like a pavement trench	something blocking the way or making it harder to pass	any anomalys, and dangerous things that could someone stumble over

Appendix 8.2-V: Q14

Q15: Were the data from the application correct?					
P1	P2	P3	P4	P5	P6
most	all	most	all	all	all

Appendix 8.2-W: Q15

Q16: What data were incorrect?					
P1	P2	P3	P4	P5	P6
little confused from the distance in the second segment	-	confusing amount of information, prefers less detailed information. Thinks the meters data in segment 4 is not correct.	-	-	-

Appendix 8.2-X: Q16

Q17: Were the instructions from the application clear and easy to follow?					
P1	P2	P3	P4	P5	P6
all	all	all	all	all	all

Appendix 8.2-Y: Q17

Q18: What instructions were not clear or easy to follow?					
P1	P2	P3	P4	P5	P6
-	-	-	-	-	-

Appendix 8.2-Z: Q18

Q19: Was the application comfortable to use on noisy and busy street?					
P1	P2	P3	P4	P5	P6
rather was	rather was	definitely was	definitely was	definitely was	definitely was

Appendix 8.2-AA: Q19

8.3 Segments of testing route

Appendix 8.3-A: Start of the segment 1

Appendix 8.3-B: Obstacle in the middle of segment 2

Appendix 8.3-C: First pedestrian crossing betweens segments 2 & 3

Appendix 8.3-D: Segment 3; Participants were asked about slope of the pavement at the end of this segment

Appendix 8.3-E: An obstacle at the end of segment 3

Appendix 8.3-F: Second pedestrian crossing between segments 3 & 4

Appendix 8.3-G: An obstacle at the beginning of segment 4

Appendix 8.3-H: Segment 4

Appendix 8.3-I: Destination; "Na Zděraze 13"

Appendix 8.3-J: After the destination street continues to an intersection

8.4 Submited files and archives

There is a list of directories, files and archives I submitted together with this bachelor's thesis. Make sure you read readme files in each particular directory for launching instructions.

1. **naviterier_appsource** - is a directory containing all source files
2. **naviterier_hybrid** - is a directory containing the hybrid wrapper
3. **storyboards** - is a directory containing all storyboards in full resolution
4. **mockups** - is a directory containing all mockups in full resolution
5. **low_fi** - is a directory containing all low-fidelity prototypes
6. **high_fi** - is a directory containing a high-fidelity prototype
7. **screens** - is a directory containing screenshots of all application screens
8. **logs** - is a directory containing all application logs from the testing

Appendix 8.4-A: Submitted content