UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Information Technology specialty

Triin Uudam

Vehicle Mileage Log Application

Bachelor Thesis (6 EAP)

Supervisor: Satish Srirama, PhD
Supervisor: Eero Vainikko, PhD

Author: ..... "....." May 2012
Supervisor: ..... "....." May 2012
Supervisor: ..... "....." May 2012

Allowed to defense
Professor: ..... "....." May 2012

TARTU 2012

Table of Contents

List of Figures

1. Introduction

The purpose of this bachelor thesis is to develop an application for iPhone smartphone that has the functions of a vehicle mileage log. The idea of a vehicle mileage log is to track all travels made by employer's vehicle and save the odometer's current indication and generate reports therefrom. It also distinguishes business and personal travels. It is mandatory to keep a vehicle mileage log for a person who is using an employer's vehicle for making private and business travels in Estonia.

Today, iPhone is popular among smartphone users. A smartphone has extra features that distinguishes it from typical mobile device. It has an operating system software that runs applications made by developers [2]. In April 2011, Apple's iOS (mobile operating system) had 28% market share of global smartphones [3]. iPhone users download and use applications from the App Store. The Apple App Store is platform for distributing applications that run on iOS [5]. In March 2012, there were about 585,000 applications available at the App Store and 25,000,000,000 applications downloaded from there [6].

This application needs global positioning system for tracking travels. A global positioning system (GPS) can be used for positioning mobile objects. It can receive radio waves from satellites and derive present position of the mobile object from them [7]. iPhone is a smartphone that has built-in GPS. Because of that there is no need to install additional GPS devices in the vehicle for tracking and saving travels. This application is currently unique in Estonia. There are similar applications available at the App Store but none of them can be directly used as a vehicle mileage log. An alternative is to use a special GPS device that saves information about made travels but it is more expensive. It has additional costs like monthly maintenance and buying or renting the device. This application has no extra costs and it meets the requirements of the form of vehicle mileage log used in Estonia. It can be used to submit the information about made travels to Department of the Treasury.

The potential users for this application are small companies whose employees use iPhone smartphone daily. This application helps to save money from more expensive GPS devices that need periodical maintenance.

This bachelor thesis describes developing the application and describes the components of it.

Chapter 2 consists of state of the art. It describes what is the purpose of the vehicle mileage log application and what kind of similar applications are available at the App Store at the moment. It also describes iOS development and Objective-C language used for developing.

Chapter 3 describes the requirements and features of the application. It also describes how the application is developed and contains screen shots from the application itself.

Chapter 4 is usability analysis. It analyzes the feedback from users and gives overall statistics.

Chapter 5 gives a list of improvements to be done to the application in the future.

Chapter 6 gives an overview and summarizes the work of this bachelor thesis.

Chapter 7 is a resume in Estonian.

2. State of the Art

This chapter contains state of the art that describes the application's essence and compares similar products to it. The chapter also includes an overview of iOS development: the platform, frameworks and Xcode. It also has a short description of Objective-C that is the language used for developing iOS applications.

2.1 Similar applications

The vehicle mileage log application is similar to digital tachograph system. "A digital tachograph system includes a vehicle unit operative to detect vehicle performance characteristics such as vehicle speed, elapsed trip distance, engine rpm, total engine revolutions, total fuel consumption, rate of fuel consumption and the like as a function of time. The foregoing vehicle performance characteristic data is displayed in the vehicle unit and is stored in a data memory located therein." [8].

This application detects only the information needed for keeping a vehicle mileage log. It saves elapsed trip with its attributes such as total length, odometer's indication in the beginning and in the end, private/business matters. The information is shown on phone's display. The data about made trip can be inserted manually or tracked by GPS function. Global Positioning System can provide accurate and continuous position information with right equipment. It gets informations from 24 satellites in 6 orbital planes, each carrying 4 satellites. The service can be used by unlimited number of users [9]. The vehicle mileage log application uses GPS built in iPhone to track user's trips in real time.

There are two applications that can be found in iTunes store with the search word "tachograph". The first application Canvoy is a client-server system that supports fleet management in real time. It collects time and status from the CAN protocols of the vehicles. Among other features it provides current position and status of vehicles and the activities of the drivers [10]. Canvoy application can be seen on figures 1 and 2. The second application iTacho helps to track truck driving and tachograph hours [11]. It's design and functions are simpler than the previous application's. iTacho screen shots can be seen on figures 3 and 4.

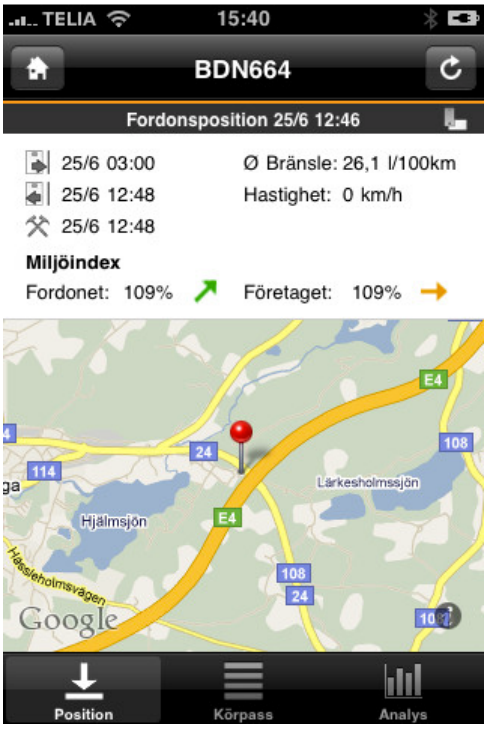

Figure 1. Canvoy application Position view

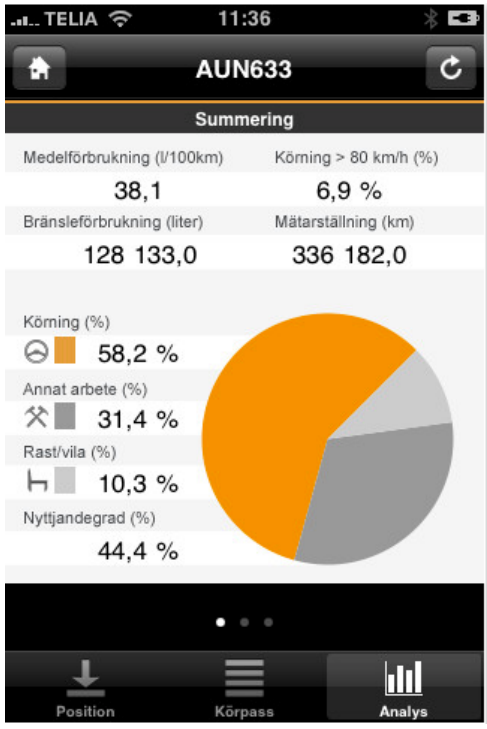

Figure 2. Canvoy application, Analysis view

Figure 3. iTruck main view

Figure 4. iTruck Records view

The travel diary application differs from previously mentioned applications. The other applications focus on overall status of vehicles and activities of drivers while the travel diary application is concentrated on monitoring specific attributes of elapsed trip.

There are several applications that can be found in iTunes store with the search words “mileage log”. Tripometer is used to keep track of user’s travel to make monthly expense report or to track mileage for income tax purposes [12]. This application has many similarities to travel diary application. The biggest difference is that the current vehicle mileage log application also has the feature of using GPS to track current trips.

Figure 5. Tripometer's Add Trip view

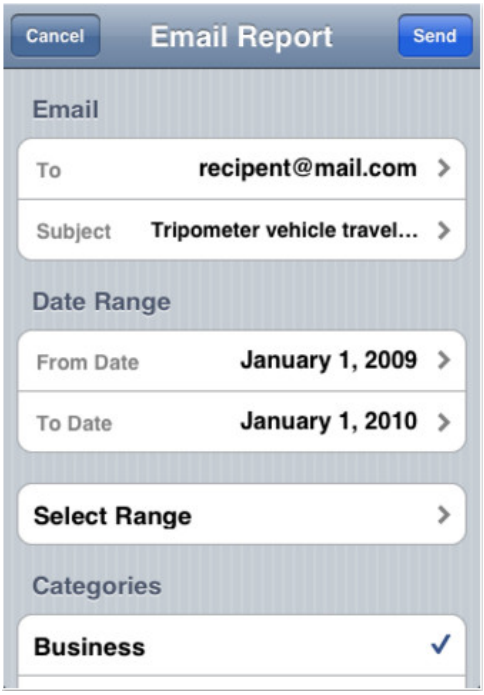

Figure 6. Tripometer's Email Report view

Overall there are no applications that match completely with the travel diary application's features. The travel diary application is based on client's needs to track elapsed trips and export monthly results of them.

2.2 iOS development

2.2.1 Introduction

The first iPhone was released in June 2007. The first version of iOS was concurrently released with the first iPhone. In June 2010, iOS 4.0 was released. It had over 1500 new APIs for developers which also included multitasking feature. "Multitasking services allow tasks to be performed in the background while preserving performance and battery life" [13]. "An application programming interface (API) is a source code-based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables." [14]. Currently in April 2012 the newest version of iOS is 5.1 [15].

2.2.2 Developing on iOS platform

iOS software development kit (SDK) and Xcode, Apple's integrated development environment (IDE) are used for developing applications. Xcode can be seen on figure 7. Among other features Xcode includes a source editor, a graphical user interface editor and source repository management. Xcode manages app projects and allows the developers to edit, compile, run and debug code. The Interface Builder tool helps developers to create user interface visually. The interface objects' library is accessible from the tool's menu on the right. It consists of many different UI components such as label, slider, switch, table view etc. The objects can be dragged and dropped to the view that's being created. Also the behavior and properties of the selected object are modifiable from the Interface Builder tool. The objects can be connected to view controller by referencing outlets. The interface objects are later loaded into the app at runtime. The Instruments tool helps to get performance analysis and debug application by gathering information about application's runtime behavior [16]. The iOS SDK extends the Xcode toolset with tools, compilers and frameworks needed for iOS [17].

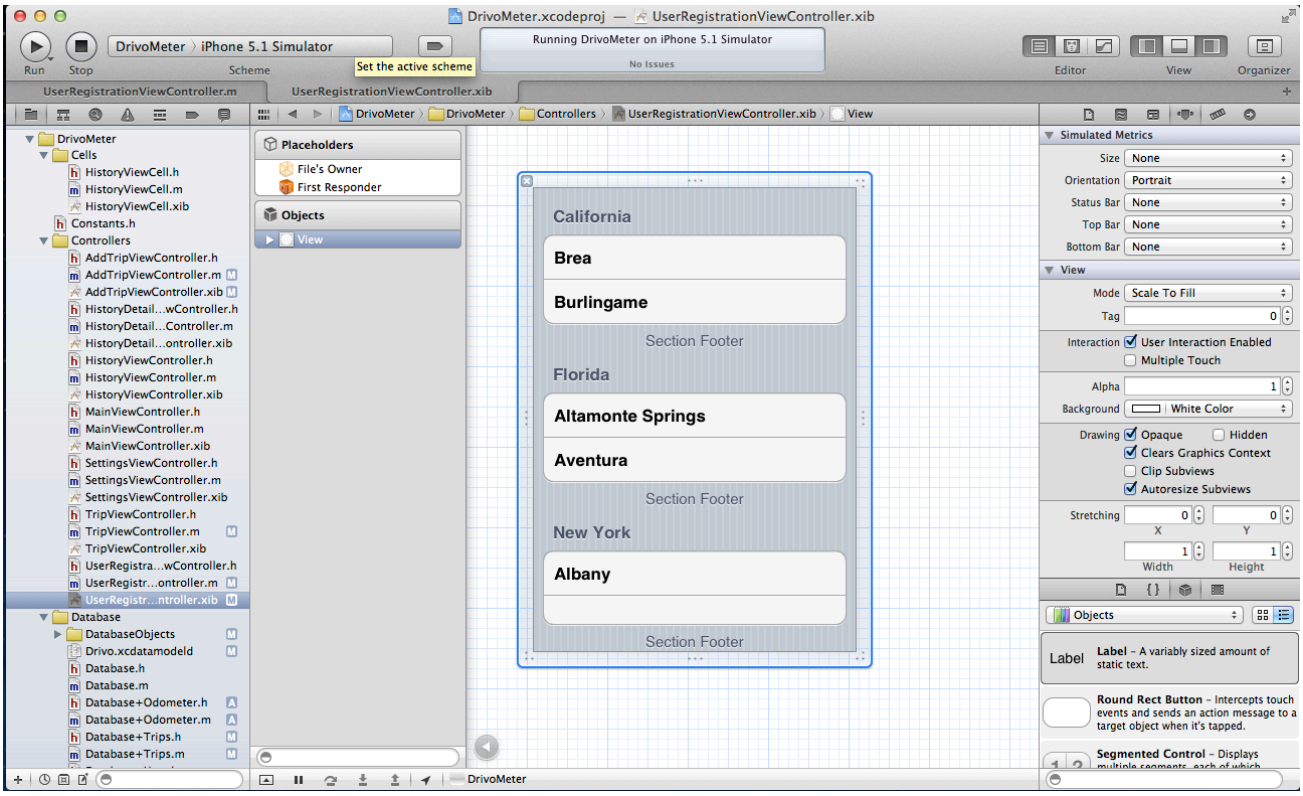

Figure 7. Xcode with the Interface Builder tool

2.2.3 Frameworks

Cocoa Touch frameworks are used for building applications. A large part of Cocoa Touch is implemented in Objective-C. It is possible to create graphical and event-driven applications with UIKit. Cocoa Touch matches with the unique interface of iOS. It is possible to access special graphical user interface controls and also the accelerometer and multi-touch gesture. The Cocoa Touch collection of frameworks also includes Objective-C frameworks such as Core Animation, Core Audio and Core Data [19].

The vehicle mileage log application uses Core Data framework. Core Data is a data-modeling framework for object-oriented Cocoa Touch applications. It provides *object-relational mapping*. It takes Objective-C objects and turns them into data that is stored in a SQLite database file and also contrariwise [20]. Core Data uses the built-in SQLite data library [21]. SQLite is not a full-fledged relational database server like Oracle, MySQL or SQLServer. Core Data helps to fetch and store data in a relational database without knowing SQL [20]. It is possible to add entities with different attributes and also make relationships between them. NSFetchedResultsController can be used to

fetch objects from Core Data. It has a NSFetchedResultsControllerDelegate that keeps the controller up-to-date with objects in database.

The second framework that the Vehicle Mileage Log application uses is Core Location framework. It can detect user's location with three technologies: GPS, cell tower triangulation and Wi-Fi positioning service. Using Core Location decreases iPhone's battery life considerably so it is recommended to use it as much as needed and no more. There is an option to specify desired accuracy for GPS. The technology that Core Location uses is hidden. Core Location will decide which technologies to use based on the accuracy needed for the request [23]. Tracking vehicle position needs the best accuracy so it drains battery and needs to be switched off whenever it is not used actively.

2.2.4 Other components of the SDK

The iPhone simulator is also a component of the SDK. It runs apps in computer in the same way as an actual iPhone device. It helps to test the application's user interface to ensure that it works correctly. Another important resource is the iOS Developer Library. It is a documentation that has all the necessary information about iOS and developing applications. There are articles, guides, sample code and technical notes in the documentation [22].

2.2.5 Objective-C

iOS applications are written in Objective-C language. "The Objective-C language is a simple computer language designed to enable sophisticated object-oriented programming. Objective-C is defined as a small but powerful set of extensions to the standard ANSI C language. Its additions to C are mostly based on Smalltalk, one of the first object-oriented programming languages" [18].

Objective-C is like the C language with some extra features. Generally classes in Objective-C have interface (.h) and implementation (.m) files. The interface file holds declarations of elements such as function prototypes, structs. It provides the public view of the class. It contains all the information necessary for someone to use the class. The implementation is the source file and contains the code that implements the methods declared in the interface. The following is an example of Hello World.m code:

```
#import <Foundation/Foundation.h>

int main(int argc, char *argv[])
{
 NSLog (@”Hello, World!”);
 return (0);
}
```

The *#import <Foundation/Foundation.h>* statement tells the compiler to look at the Foundation.h header file in the Foundation framework. The NSLog() function prints “Hello, World!” to the console [24].

2.2.6 Memory Management

One of the most important parts in using Objective-C is memory management. Objects have a life cycle. Every object has a retain count. When some part of code is interested in an object, it increases the object’s retain count. When it is done with the object, it decreases the count. When the retain count goes to 0, it is destroyed and the memory is returned to the system. Commands like *alloc, new* or *copy* increase the object’s retain count. *Release* message decreases its retain count. *Dealloc* message is sent to the object when its retain count has reached 0. There is also a method called *autorelease* that schedules a release message to be sent at some time in the future [24]. It is most helpful in methods that return newly created object but cannot release it before returning it. With the release of iOS 5.0 Apple introduced Automatic Reference Counting (ARC). It simplifies the process of managing the lifetimes of Objective-C objects. ARC evaluates the lifetime requirements of objects and automatically inserts the appropriate method calls at compile time [25]. Mostly it means that the developer cannot call *retain, release, autorelease* or *dealloc* methods in the code. It helps to avoid memory leaks and makes developing process easier for the developer. The difference between ARC and manual reference counting can be seen on the following figure 8.

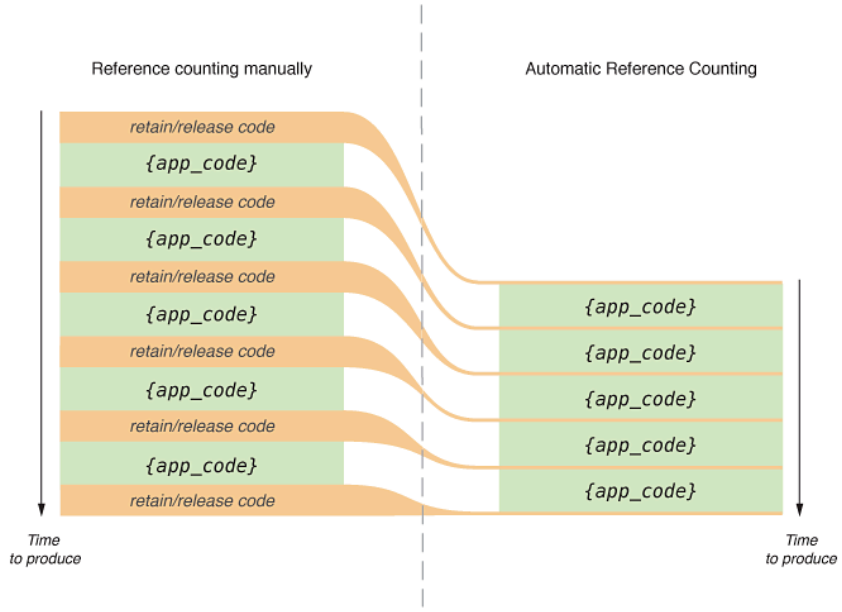

Figure 8. The difference between manual reference count and ARC

3. Designing the Application

The application has following requirements, which are explained in detail in section 2.1 (functional requirements) and 2.2 (non-functional requirements). Section 2.3 gives an overview what is used for developing the application. The application's screenshots and description are added in section 2.4.

3.1 Application functional requirements

The vehicle mileage log application has several requirements for optimal user experience.

3.1.1 GPS track for active travels

The user can start saving and tracking new active travel with one click. They can also choose whether it is a private or a business trip. The application updates information about distance traveled whenever GPS updates the location. The active travel can also be stopped with one click. The travel will be saved to local database.

3.1.2 Add new travel manually

The user can add trips manually if they need to. If there are trips made in the past that were not tracked with GPS, the user can click the button "Add" and insert the travel and its information manually. The information needed about the travel is the date the trip was made on, odometer's state after the trip, the purpose of the trip (private or business). The date is automatically set to current date but it can be changed by clicking on the date cell.

3.1.3 Delete travels

The user can delete saved travels from the history view. There is a button called "Change" in the history view and it opens up a table editing view. The user can then delete the travels they would like. There is also another way for deleting travels in the same view. If the user swipes over a cell, then the application automatically offers to delete the selected row.

3.1.4 Distinguish private and business travels

Each travel is marked as private or business. They have to be distinguished due to the regulations of Department of the Treasury. There are additional taxes for making private travels with employer's vehicle. The opportunity to distinguish travels is available for both in adding travels manually view and in currently tracked travel view.

3.1.5 Monthly results export to e-mail

The user can export monthly results from travel history to their e-mail. The result sheet includes all the needed information for vehicle mileage log (name, car number, company's name, history of travels and their properties). The results can be presented to Department of the Treasury.

3.1.6 History view of made travels

The application has a history view. It shows the information about made travels. The main history view shows whether the travel was business or private and when was it taken. There is a detail history view for every travel that shows more information. The detailed view includes showing the distance of the travel and odometer's beginning and ending indication.

3.1.7 Choose period for history view

The user can use filter on history view to see only selected period travels. There is a button called "Filter" that opens a new view for selecting history period. The user can choose this week, this month, last week, last month and all period to filter travels.

3.1.8 Favorite travels

The user can add made trips to favorites. If there are trips that are made regularly by the user, it is easier to add them manually from favorites list. This saves iPhone's battery from using GPS and also time from tracking. The user can add and manage favorite trips under extras view. It is also possible to save a trip as favorite from history details view.

3.2 Non-Functional requirements

3.2.1 Maintainability

The application software is written keeping in mind the best practices for maintainability. The code follows object-oriented design. Each component of the system is separated from the others therefore the system can be improved and modified easily. New requirements are easy to implement. Changing the code does not affect the whole system but only needed components.

3.2.2 Performance

The application needs Internet access to send trip reports to the user's email. Sending a report should take no more than 2 seconds depending on the speed of the Internet.

The application also depends on GPS system when the user is tracking a trip with GPS. The first update for user's location should take no more than 7 seconds. It mostly depends on the signal of the GPS. If the weather is fine (not very cloudy) and there are no big obstacles e.g. tunnels or skyscrapers, the system should update user's location in about 2 seconds. If the GPS signal is very weak, it may happen that the user's location will not be updated.

3.2.3 Platform

The application runs on iOS operating system. The iOS version should be at least 5.0 because the application uses some features that are not implemented in previous versions (e.g. following users location on the map with built in method). iOS 5 is compatible with iPhone 3GS, iPhone 4 and iPhone 4S [32]. Previous models of iPhone are not supported. This means that the application will run only on the three previously mentioned iPhone models. The programming language used for developing the application's software is Objective-C.

3.2.4 Usability

This application is easy to learn and use. Its design is made according to the client's needs and requirements. The application views are intuitive and have as few user interface components as needed to keep it simple to use. The navigation in the application is easy because the main views (History, Trip, Extras) are accessible all the time throughout using the application.

3.3 Methods for developing

3.3.1 Model-View-Controller

iOS development is mostly built up on the model-view-controller pattern.

3.3.1.1 The Model

The model is used for storing data. It is totally independent and has no references to view and controller. The model objects can be used anywhere in the project for storing and fetching data. It is important that the model does not know anything about views and controllers. This way it can be used by different controllers and views. The Vehicle Mileage Log application uses Core Data to hold model objects. It has three entities with different attributes. Views and controllers can use them to show information [26].

3.3.1.2 The View

The view consists of user interface elements such as buttons, user inputs, table views. It is used for showing information to user. It is tightly connected to the controller but is still a separate component in the MVC pattern to hold interface elements and show data. In Xcode the views can be created with interface builders [26].

3.3.1.3 The Controller

The controller is the key system between the view and model. It is responsible for showing correct information in the view and also updating the model, if necessary. It can also fetch objects from model and get the latest updates to show in the view [26].

3.3.2 Application's User Interface

The application is designed by following iOS Human Interface Guidelines. The guidelines give an overview of most used user interface elements in UIKit framework. The user expects a certain behavior from views that are also used in built-in applications therefore it is good to know how and when these UI elements should be used in the application.

3.3.2.1 Navigation bar

Each view has a navigation bar at the upper edge of the screen. The navigation bar always displays the title of the current screen. If needed, it also has a back or a cancel button on the left side and a save, filter or manage button on the right side. There is never more than one button on each side of the title.

3.3.2.2 Tab bar

There is a tab bar at the bottom of the screen and it can be used from every view. The tab bar has icons for each tab that give the user access to the three main view: history on the left, trip in the middle and extras view on the right. The icons have been chosen according to the functions and meaning of the view.

3.3.2.3 Table view

A table view is used in different views to present data. It is used for adding new trips, showing history, trip details and favorite trips, configuring user and GPS. Rows in table view have disclosure indicators if they need to display another associated view. Rows have checkmarks if they need to indicate current selection in table view. If the user clicks on a row, it responds immediately with necessary action.

3.3.2.4 Other components

An action sheet is displayed when the user uses GPS tracking the first time. It is also displayed with options to save, continue and cancel if the user clicks the stop button for currently tracked trip.

A segmented control is used for selecting private of business attribute in adding new trip view. The segments have short understandable titles and are easy to tap.

A switch is used in GPS settings view to select GPS navigation accuracy and to turn the displaying of map view on or off.

Text fields are used in table views so the user can add or change information. [27]

3.3.3 External classes

The Vehicle Mileage Log application uses several external classes for implementing different functions. These classes help to make developing easier as they provide different specific methods needed for the application. All the following classes are open source and free to use.

3.3.3.1 MBProgressHUD

MBProgressHUD class provides a simple progress hud for view that is completing some task. It can be added to view to show user that some work or executed request is in progress. MBProgressHUD also allows to add text under hud to show additional information to the user. MBProgressHUD can be added to view with [MBProgressHUD showHUDAddedTo:self.view animated:YES] method and removed from view with

[MBProgressHUD hideHUDForView:self.view animated:YES] method. It can also be used for informing the user that some task has been completed with a checkmark hud view [28]. The current application uses a indicator hud on the map view before it starts updating the user's location. It also uses a checkmark hud view after the export report is sent to email.

3.3.3.2 SKPSMTPMessage

SKPSMTPMessage class implements a SMTP client for the application. With this class the application can send export reports directly to user's email without showing user the MFMailComposeViewController. This helps to save time for users because they don't have to send the email themselves but the application sends it automatically. SKPSMTPMessage object has to be configured with the email that the export data is sent from (host, login name, password, subject, attachments, content) [29].

3.3.3.3 CHCSVParser

CHCSVParser class is used for parsing CSV files in Objective-C. The Vehicle Mileage Log app uses CHCSVWriter class that helps to construct CSV files. It has different methods for writing lines and fields. Method -*writeLine* starts a new CSV line. Method –*writeFields* takes an array of strings and writes new CSV field with each string. The application uses CHCSVWriter for parsing selected trips' information to CSV file [30].

3.3.3.4 BreadCrumb

BreadCrumb is a sample code provided by Apple. It demonstrates how to track user's current location and draw a path to illustrate it on the map view [30]. The application uses it to display user's current location and traveled distance on the map if he or she uses GPS tracking for making a new trip.

3.4 Application views

3.4.1 Screenflows

The following figure no. 9 shows the application's screenflow. The main three views that are on the tab bar are with beige background. There is an abstract main view controller with blue background that creates the tab bar and holds the main three view controllers. Each box represents a view in the application. All the actions that can be done from the view are also listed under the view's title. The arrows show from where

the view can be accessed. Some views can be accessed from multiple views. For example User Information View is shown from the Main View the firs time application is launched. After that it can be accessed from Settings. Filter View is used from History and Export views for selecting period. Favorites View can be accessed from Adding Trip Manually and Extras View.

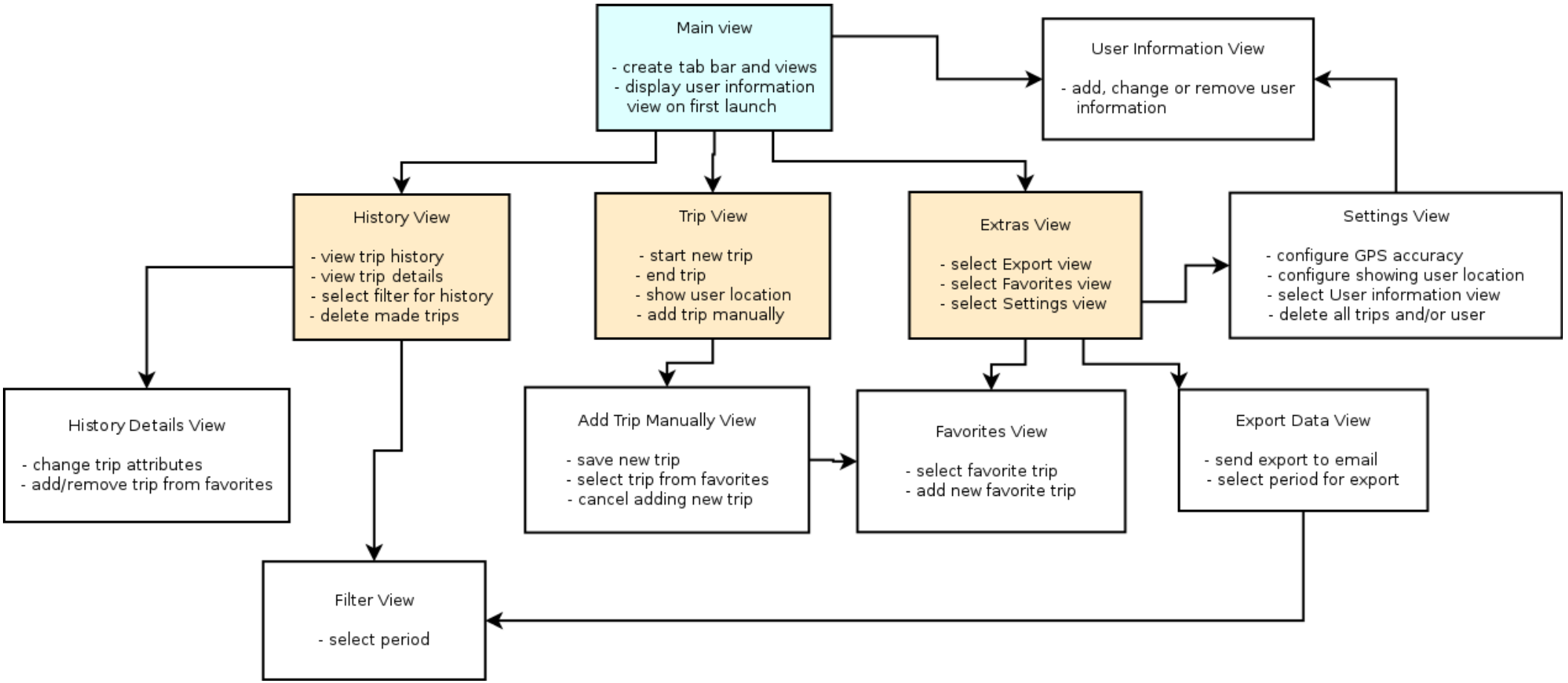

Figure 9. Screenflow diagram for Vehicle Mileage Log application

3.4.2 Application views

The first and main view of the application is GPS tracking with map view (figure 10). This is the main view because the user should be able to start a new trip as easily as possible. This is also the main function of the application. The Start button starts a new trip that is tracked with GPS. Concurrently with tracking there is a map view that can show user's location and travelled distance. After every GPS update the map overlay is updated and a point is added to the line that shows user's travelled distance.

Trips can also be added manually. This can be done by tapping the button with plus sign on the right side of the navigation bar. New view will open with distance and purpose properties. The view can be seen on figure 11. The odometer's indication is automatically updated after the distance is inserted.

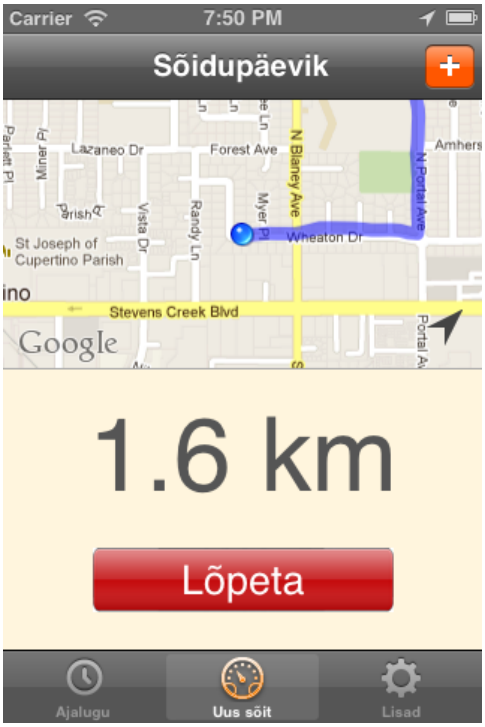

Figure 10. Main view with GPS tracking

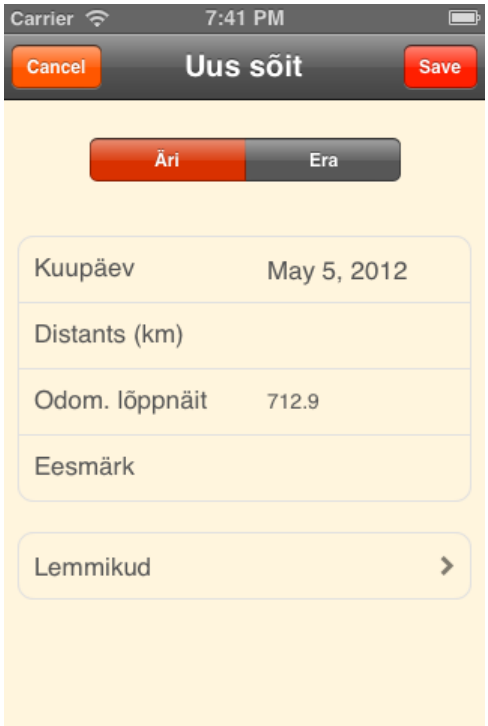

Figure 11. Add new trip manually

The second view on the tab bar (on the left) is History view (figure 12). It shows trips made by the user. Custom cells are used for showing information about trips. Most important information about the trip is shown in the history view. The letters "E" and "Ä" indicate whether the made trip is private or business. The title shows the purpose of the trip. The subtitle shows the day and date of the made trip. There is a star shown in

the cell if the trip is also a favorite. If the user clicks on a cell, Detail History view will open. This view shows detailed information about the trip, including odometer's beginning and ending indications. The view can be seen on figure 13. The trip can also be edited from this view. If the user wants to change trip's attributes he or she can do this and save the changes by pressing the "Save" button. The trip can also be added or removed from favorites in this view.

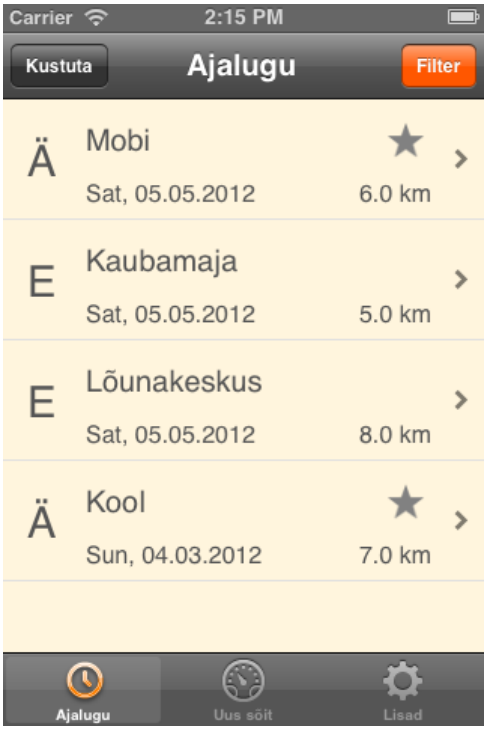

Figure 12. History view

Figure 13. Detail History view

The third view on the tab bar (on the right) is Extras view that is seen on figure 14. The Extras view handles additional features of the application. From here the user can navigate to Export view (figure 15). The trips' information can be exported to user's email in CSV format. The user should insert their email address and the period between the exported trips are made. The information is exported to CSV file using *CHCSVWriter* library and then emailed from the Vehicle Mileage Log applications's email using *SKPSMTPMessage* library.

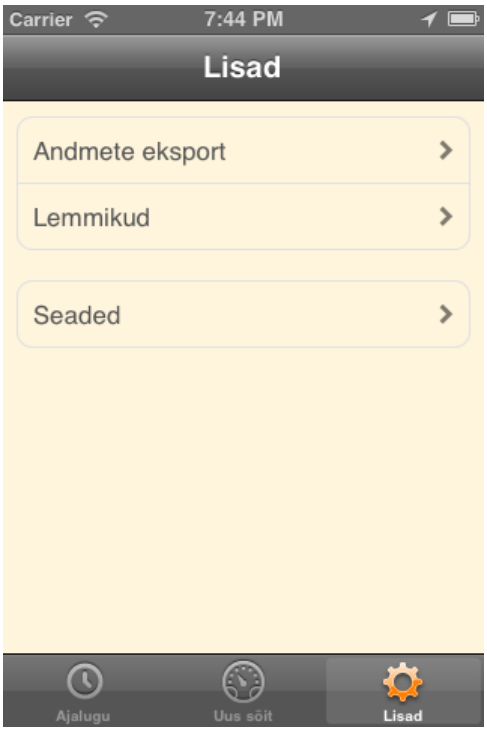

Figure 14. Extras view

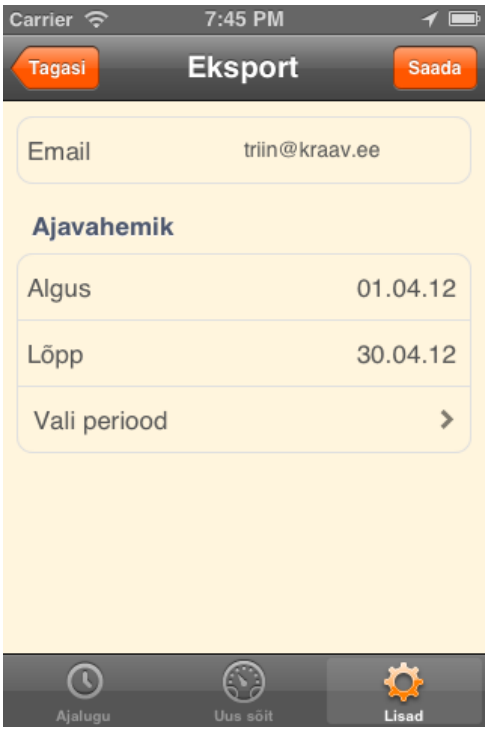

Figure 15. Export view

The Favorites view is used for trips made regularly. The view can be seen on figure 16. The user can add his or her trips to favorites and later use them if trips are added manually. This makes adding trips easier because the user does not have to insert the same information every time again.

The Settings view (figure 17) displays different options for the user. The user can configure the GPS signal quality. Better quality means draining more phone's battery. The user can switch to less accurate GPS signal to save battery. By default the GPS is configured to best accuracy for navigation.

The user can also configure whether to show user's location on the map. Showing user's location uses data communication or Wi-Fi for updating the map view. If the user does not want to use it, they can turn it off.

The user can navigate to User Information from the Settings view and change or add the information, if necessary. The user's information can also be deleted from the settings menu. The user can choose to delete all trips or user information.

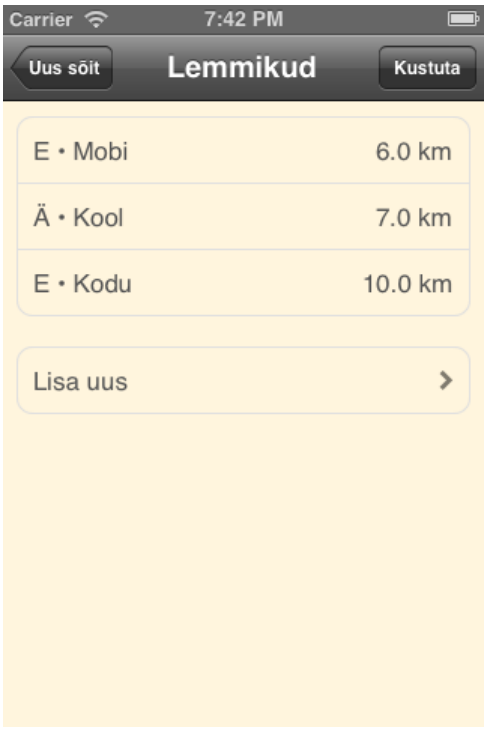

Figure 16. Favorites view

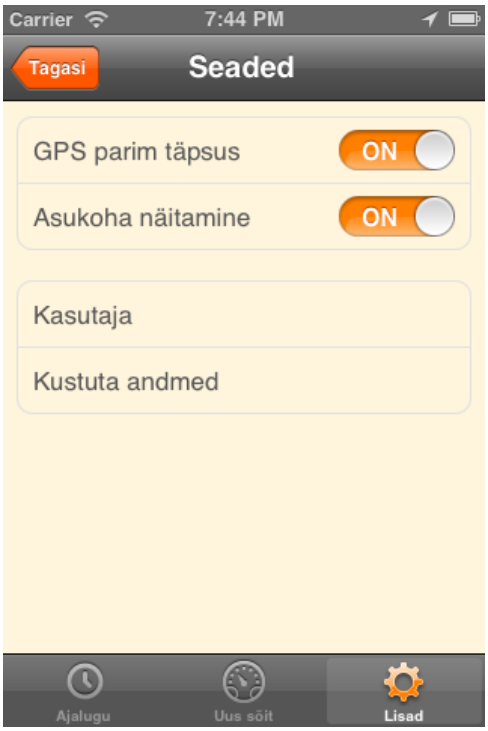

Figure 17. Settings view

The User Information view (figure 18) displays all the information inserted about the user. It is possible to change and save the information in this view. The user view also appears if the application is opened the first time and there is no information about the user. In general it can be viewed from the Settings.

The Filter view (figure 19) is used by History view and Export view. History uses filter to display made trips in the selected period. Export uses filters to export information in given period. If the period is selected, the Filter view automatically pops to previous view.

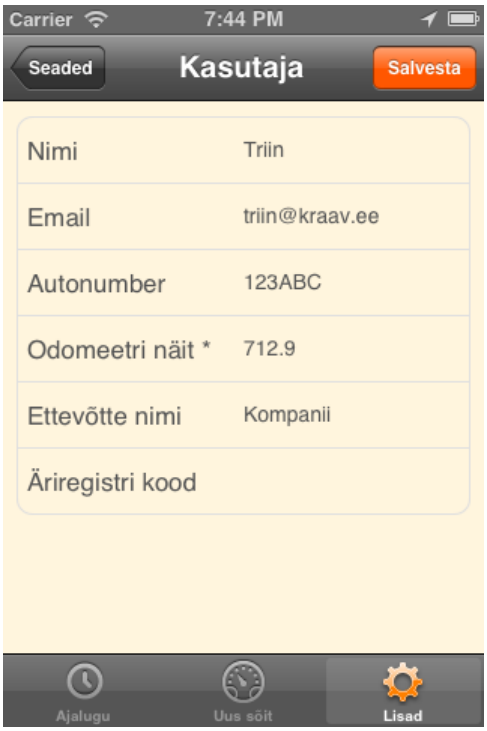

Figure 18. User Information view

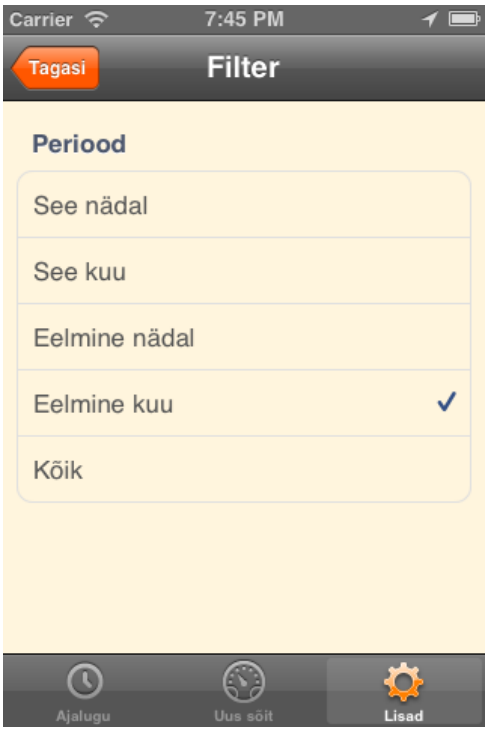

Figure 19. Filters view

4. Analysis of the Application

The following chapter describes how the application was tested and what was the outcome. Section 3.1 gives an overview of the environment the application was tested in. Section 3.2 analyzes the results of testing. The application code is available in Appendix B.

4.1 TestFlight

TestFlight is an environment for testing iOS applications. It can be seen on figure 20. Developers can upload application builds to TestFlight for testing. They can invite users to become testers. Once the invitation is sent, the tester registers as a TestFlight user and then his or her device's unique identifier number is sent to the developer. The developer then can add the device to provisioning profile to grant access for the tester. After the permission is granted, the tester can download the provisioning profile for the device and after that the application for testing through TestFlight app.

TestFlight shows information about testing for developers. It is possible to download and integrate TestFlight SDK to the tested application. If the application is integrated, it can send information about testers' activities, sessions, crashes and checkpoints. It is also possible to ask questions from testers in the application during the testing. TestFlight SDK enables developers to add checkpoints to code wherever needed. If the tester passes checkpoint in the application, a notification is sent to TestFlight. The developer can then analyze the using of the application. For example a checkpoint about adding new trip manually can be added. If the user presses Save button in the application to save new manually added trip, the developer receives notification about it in TestFlight.

It is also possible to upload new application builds and notify testers. The testers can then install the new version and start testing. TestFlight shows also information about what kind of device is each tester using, what is the last version installed and when is it updated.

TestFlight is useful for getting information about testers and their testing results. It can also make testing easier for testers who are not in the same region with developers. They can access the application that needs testing remotely [33].

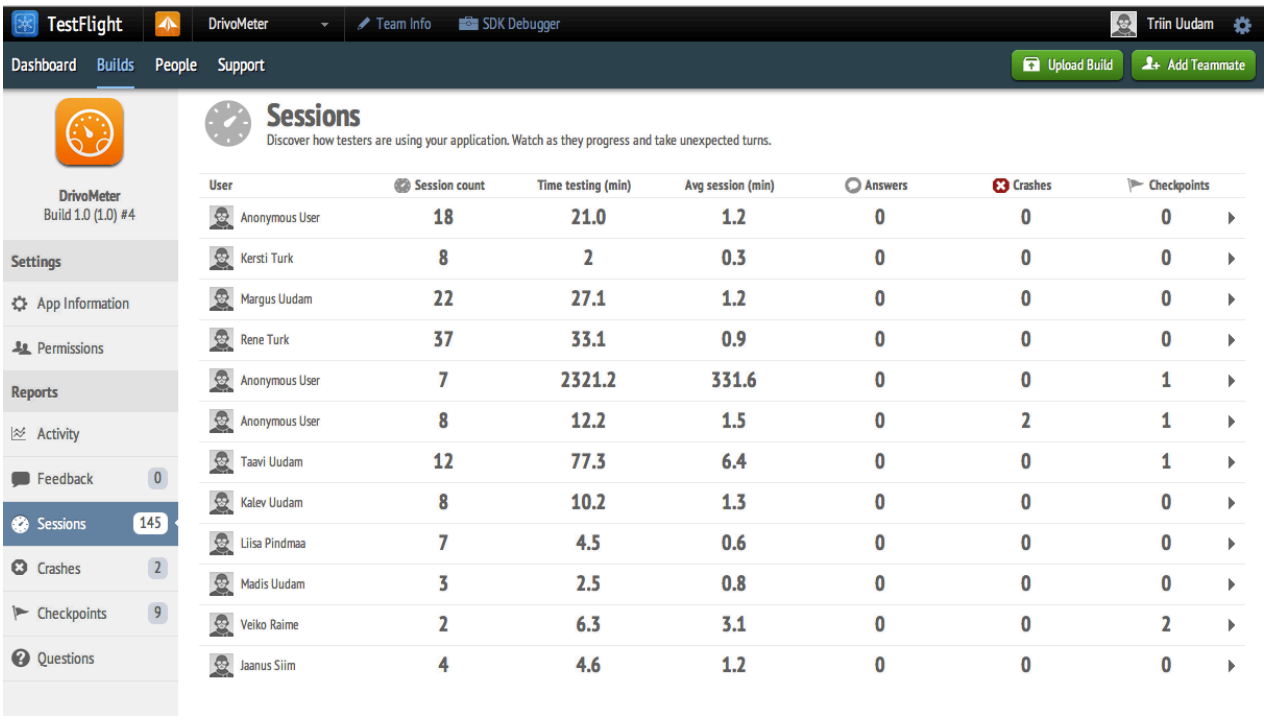

Figure 20. TestFlight environment

4.2 Conducting the survey

The developed application was tested by 10 iPhone users. The testers were aged between 12-65. The application was tested on iPhone 3GS, iPhone 4 and iPhone 4S. The iOS versions on the devices were 5.0, 5.01 and 5.1. The application was tested using TestFlight environment and application. The testers were given the Vehicle Mileage Log application and a questionnaire. The survey consisted of 21 questions, including 5 tasks. The user had to complete tasks and after that had to answer how difficult the given task was. The questionnaire can be viewed in Appendix A.

There are additional statistics from TestFlight environment about the testing. There were a total of 139 sessions started by testers during the testing period. The minimum number of sessions done by one tester was 2. The maximum number of sessions by one tester was 32. The time the application was tested during one session varied from 2 minutes to 77.3 minutes. A total of 8 checkpoints were passed in the application, including sending email, changing trip's attributes, saving trip manually. There were a

total of 2 crashes for the application during the testing sessions. Both crashes were accomplished by one tester. The crashes were related to incorrect handling with UITextField resigning first responder and showing custom view with date picker on table view.

4.2.1 Adding new trip

The first task was to add a new trip manually. This task was chosen because it is one of the main functions of this application. After the task the testers had to answer how easy the given task was. They had to choose among 6 answers: very easy, easy, medium, hard, very hard and could not fulfill the task. 70% of testers responded that the given task was very easy and 30% responded that it was easy. No other answers were used. No remarks were made about this task. It can be said that overall adding new trip manually is very easy to execute for users.

4.2.2 Change trip from private to business

The second task for the testers was to find a made trip from history, select trip details, click on the cell that shows private/business value and save changes. This was selected as a task to see if the users can find trip details from history and change them if necessary. After completion the testers had to answer how easy it was. The options were very easy, easy, medium, hard, very hard and could not fulfill. 60% of testers answered that the given task was very easy. 40% answered that it was easy.

Additional question about the task was that if it seemed hard, how would the testers change it. Four of the testers answered that although the task was easy to complete, the screen had no hints that the private/business value could be changed by clicking on the cell. It was suggested to add an explaining user interface element that helps the users understand that the value can be changed.

4.2.3 Track new trip with GPS

The third task was to start a new trip with GPS tracking. The testers had to answer how easy ending and saving additional information about the trip was. 90% of testers answered that it is very easy. 10% found the task to be easy.

Additional question was about the third task was that what are good and bad sides of current implementation of tracking trip with GPS. There were several only positive answers saying that it is easy, intuitive, clear. The testers also liked seeing user's location on the map view. On the other side it was added that the data communication is

quite large when using map view. A test drive was made that showed 787 KB of used data for a 9 km trip. This means that the user should have a limitless data communication to use the showing user location feature. The map view can be turned off and due to large data communication it should off by default. There was a comment about turning the map view setting off in the middle of an active trip. Currently the map view will be not used from the next tracking. It should be turned off right after the user has chosen that setting.

It was also mentioned that a trip's purpose is asked after finishing a private trip. It is not necessary as the purposes are mandatory for only business trips.

Nevertheless there were three testers who said they would not change anything with the current tracking implementation.

4.2.4 Adding trip to favorites from history

The fourth task of the survey was to add a trip from history to favorites. It was added as a task to see how the testers understand adding trips to favorites and using them. 90% of testers found this task to be very easy. 10% of the testers thought the task was with medium difficulty. Overall it can be said that it is very easy for users to add a trip to favorites. The additional question to the testers was what would they change with adding a trip to favorites. There was a suggestion to separate the favorite row from others with a section to make it more understandable and noticeable to users.

The next question was how informative the overall history view looked. Three testers answered that it looked good. There was a suggestion to also add a map view with the trip's trajectory to History Details view.

4.2.5 Export data

The fifth task was to export data to email. This is one of the main and most important functions of this application and therefore was selected as a task for testers. 80% of the testers found this task very easy. 20% of the testers answered that it was easy. This shows that overall the users can export data very easily. There was additional question what should be changed with the export. It was suggested that since exporting data is one of the main functions of this application, it should be moved from Extras view to History view. One of the testers did not know that the e-mail field should be filled in order to send data.

4.2.6 Overall evaluation for the application

The final step of the survey was to evaluate application's different aspects. The first question was to evaluate the application with following criterions: intuitiveness, user friendliness, application's design, how easy it is to study and how easy it is to use the app. Each criterion had a value from 1 to 10. The following figure 21 illustrates the results.

From the figure it can be seen that intuitiveness was rated with a total of 89 out of 100 whereby 70% of the testers found intuitiveness worth 9 or 10 out of 10. 10% of the testers found intuitiveness to be 7 out of 10. This was also the lowest score for given criterion.

User friendliness was rated with a total of 90 out of 100. 30% of the testers rated user friendliness as 8 out of 10. This was the lowest score as well. Therefore the application can be considered user friendly.

The application's design was rated with a total of 93 out of 100. The lowest score for the design was 7 out of 10. 60% of testers rated the design 10 out of 10.

The testers considered the application to be easy to learn. The total score for this criterion was 92 out of 100. 10 % of testers rated it as 7 out of 10. This was the lowest score for given criterion as well.

It can also be said that the application is easy to use. This criterion got the result of 94 out of 100. The lowest score for this criterion was 8 out of 10. 50% of the testers rated it as 10 out of 10.

	1	2	3	4	5	6	7	8	9	10	Responses	Total
Intuitiivne	0%	0%	0%	0%	0%	0%	10.00%	20.00%	40.00%	30.00%	10	89
Kasutajasõbralik	0%	0%	0%	0%	0%	0%	0%	30.00%	40.00%	30.00%	10	90
Disain	0%	0%	0%	0%	0%	0%	10.00%	10.00%	20.00%	60.00%	10	93
Lihtne õppida	0%	0%	0%	0%	0%	0%	10.00%	10.00%	30.00%	50.00%	10	92
Lihtne kasutama hakata	0%	0%	0%	0%	0%	0%	0%	10.00%	40.00%	50.00%	10	94

Figure 21. The results of the Vehicle Mileage Log application's evaluation

The next question for evaluating Vehicle Mileage Log was to answer what are the positive sides of this application. The testers thought the application to be very easy to use, logical, with nice design, practical, easy to learn. It was also thought to be positive that the application makes keeping track of trips easier because there is no need write anything down by hand.

The next question was about the negative sides of this application. It was pointed out that the application is missing a user guide or help. It was also considered as negative that the application does not support earlier versions of iOS before 5.0.

The testers also answered to a question, what would they change in this application. It was suggested to bring the export option to history view because it is one of the main functions and should not be under extras. There was also a suggestion to replace the word favorites with the word regulars. That was explained by the word favorite being awkward to use for business trips. Another suggestion was to add speedometer to the main view if the trip is active and showing user's location is turned off.

4.2.7 Conclusion of the survey

Conducting the survey among testers was a success. In conclusion the application received good feedback. It also received some remarks and comments how to improve in the future. The application's client was also one of the testers. He is pleased with the results. Therefore the development of this application has fulfilled its purpose.

It is also positive that during the testing the application crashed only 2 out of 132 times. This is 1.5% of the tested sessions. It shows that the application is also quite stabile.

5. Future work

Taking into account the testers' suggestions, following modifications should be done to the application in the future. This should improve user experience and also make using the application easier and more understandable.

- Export option should be moved under History view. This is due to the fact that exporting data is one of the main functionalities in the application and therefore should not be under extras.
- The expression "Favorite trips" will be replaced with the expression "Regular trips". This is because the word favorite may not be suitable for example business trips made often. The word "regular" is more neutral and can be used for both private and business trips.
- If the function for showing user's location is turned off, the main view should show speedometer instead of map view. The map view has no use when it does not show user's location. Instead the application could display the current speed of the vehicle.
- There should be a setting for the user to choose whether they want to insert the purpose for a private trip. It is not required by the law. If the user chooses not to insert the purpose of a private trip, the application should not display the insert purpose alert after finishing the trip with GPS tracking.
- An alert should be displayed for the user if the tracking function is left on and the user has not used it for two hours. If the distance has not been changed during two hours and the speed of the vehicle is 0, the user should be reminded that the application is still tracking the trip's distance.
- The user should be able to see the distance of the trip on the map view under History Details view.
- The user should be able to choose to export only private, business or both types of trips. It should be an option because some of the users may only want to export private trips.

6. Conclusion

The purpose of this bachelor thesis was to develop an application that fulfills the purpose of vehicle mileage log. This purpose has been fulfilled as the application is ready and the client is satisfied with the results. The application implements all the requirements needed. The main functions are starting a new trip with GPS tracking, adding trips manually, showing history and exporting trips' information to email. There are several additional features that make using the application better for the user. For example favorite trips that are made on regular basis, can be saved and later used for adding trips manually. When tracking a trip with GPS, the user can also see his or her location from the map. It can also be turned off since this feature requires data communication.

The application is developed on iOS platform. Xcode program and Cocoa Touch framework are used for developing. There are also several external classes used in the project that made developing easier. For example hud with loading indicator classes for showing that the app is working on some progress. Another example is CHCSVWriter that helps to export selected trips' information to CSV format.

The application is designed following Apple's Human Interface Guidelines. This means that an iPhone user expects some type of behavior from interface objects that are built in. This makes learning the application easier for the users.

The application was tested by 10 iPhone users. They answered a survey that consisted of 5 tasks, questions about completing them and general questions of using the application. Overall the results were very positive as the testers rated the app to be intuitive, user friendly and easy to use. There were many suggestions from the testers for improving the application in the future. Many of them are taken into consideration to make the application even easier to use and to improve the user experience.

7. Resümee

Käesoleva bakalaureusetöö eesmärgiks oli valmis arendada rakendus iPhone'i nutitelefonile, mis suudab täita sõidupäeviku pidamise funktsioone. Eestis on inimestel, kes kasutavad sõitude tegemiseks tööandja sõidukit, kohustuslik pidada sõidupäevikut. Antud rakendus muudab selle pidamise lihtsamaks. Sõite saab sisestada, kasutades *GPS* reaalajalist jälgimist või lisades sõite käsitsi. Sõitude kokkuvõtteid saab eksportida kasutaja emailile. Lisaks eelnevatele põhifunktsionaalsustele on rakendusel ka erinevaid lisavõimalusi, näiteks sõitude lisamine lemmikutesse. See võimaldab sõitude täitmist lihtsustada, kasutades juba eelnevalt salvestatud malle.

Antud rakendus on arendatud iOS platvormile, kasutades Xcode arenduskeskkonda. Rakenduse arendamisel on kasutatud erinevaid iOS platvormi raamistikke, lisaks veel mitmesuguseid avatud lähtekoodiga projektide klasse. Lisatud klassidel on erinevad funktsioonid, mis lihtsustasid antud rakenduse arendamist. Rakenduse kujundamisel on lähtutud Apple'i kasutajaliidese arendamise juhtnööridest. See võimaldab iPhone'i kasutajatel rakenduse kiiremini selgeks õppida, kuna rakenduse käitumismustrid on juba varasemast selged. Rakendus on selge ja lihtsasti kasutatav. Sellel on kolm põhivaadet: ajalugu, uus sõit ning lisad. Kasutaja saab vaadata sooritatud sõite ajaloo vaate alt. Igal sõidul on ka detailvaade, kus kuvatakse täpsem informatsioon. Uue sõidu vaates saab lisada sõitu käsitsi või alustada uue sõidu jälgimist *GPS*-ga. Lisade vaates saab navigeerida lemmiksõitude, andmete eksportimise või seadistuste vaatesse. Rakendusse saab sisestada lemmiksõite. Neid kasutades saab regulaarselt tehtud sõite hõlpsamalt sisestada, kuna nõutud väljad uuele sõidule täidetakse kasutaja eest ära. Eksportimise vaates saab valida perioodi eksporditavate sõitude tarbeks ning saata vastava kokkuvõtte kasutaja *e-mailile*.

Rakendust testis kokku 10 iPhone'i kasutajat. Testimine viidi läbi *TestFlighti* keskkonnas, mis aitab rakendust hõlpsalt testimiseks välja jagada. Läbi *TestFlighti* oli võimalik saada ka mitmesugust informatsiooni testijate tegevuse ning statistikat rakenduse kasutamise kohta. Sealhulgas on teada, et kokku käivitati testimise ajal 139 sessiooni, nendest 2 juhul lõppes testimine kokku jooksmisega. Üleüldiselt jäid testijad rakendusega rahule. Hinnangud rakenduse intuitiivsusele, kasutajasõbralikkusele,

disainile ning kasutamise lihtsusele olid väga kõrged. Seega võib öelda, et rakendust on lihtne õppida ning kasutama hakata.

Testijatelt sai mitmesugust tagasisidet ja ka soovitusi tulevaseks rakenduse edasiarenduses. Nende seas leidus mitmeid häid ideid, mille lisamine teeks kasutamiskogemuse kasutajale paremaks. Seega kuigi rakendus töötab juba antud hetkel stabiilselt ning kasutajale mugaval viisil, on tulevikus siiski plaanis seda edasi arendada.

Bibliography

[1] - [http://en.wikipedia.org/wiki/IPhone] 25.03.2012

[2] - [http://www.phonescoop.com/glossary/term.php?gid=131] 25.03.2012

[3] - Munchback, A. “Android grabs 53% of global smartphone market share; iOS 50% of application revenues” [http://www.bgr.com/2011/05/19/android-grabs-53-of-global-smartphone-market-share-ios-50-of-application-revenues/] - 25.03.2012

[4] - [http://en.wikipedia.org/wiki/IOS] 25.03.2012

[5] - [http://en.wikipedia.org/wiki/App_Store_(iOS)] 25.03.2012

[6] - [http://en.wikipedia.org/wiki/App_Store_(iOS)] 25.03.2012

[7] - Masumoto, Y: “Global positioning system,” 1992 http://www.google.com/patents?hl=en&lr=&vid=USPAT5210540&id=_VsdAAAAEBAJ&oi=fnd&dq=global+positioning+system&printsec=abstract#v=onepage&q=global%20positioning%20system&f=false 25.03

[8] - Weisbart, Emanuel S. “Digital tachograph system with digital memory system” [http://www.google.com/patents?hl=en&lr=&vid=USPAT4188618&id=UKt7AAAAEBAJ
&oi=fnd&dq=tachograph&printsec=abstract#v=onepage&q=tachograph&f=false] 11.03.2012

[9] - Kaplan, Elliott D., Hegarty, Christopher J. “Understanding GPS: principles and applications,” page 3 [http://books.google.ee/books?hl=en&lr=&id=-sPXPuOW7ggC&oi=fnd&pg=PR7&dq=GPS&ots=2r2uBALMkI&sig=V7Xq-6GReaUl6T8fRKZiuevO6iU&redir_esc=y#v=onepage&q&f=false] 11.03.2012

[10] - [http://itunes.apple.com/ee/app/canvoy-client/id381787166?mt=8] 11.03.2012

[11] - [http://itunes.apple.com/ee/app/itacho/id477697209?mt=8] 11.03.2012

[12] - [http://itunes.apple.com/ee/app/tripometer-mileage-log/id286595495?mt=8] 11.03.2012

[13] - [http://www.webcitation.org/5oqYO6thc], 25.03.2012

[14] - [http://en.wikipedia.org/wiki/API] 25.03.2012

[15] - [http://en.wikipedia.org/wiki/IOS_version_history], 25.03.2012

[16] - Baddam, L: “The iPhone Operating System (iOS) for Developers” [http://www.ece.gatech.edu/academic/courses/ece4007/11spring/ECE4007L05/kj8/iOSSDK_TRP.pdf] 25.03.2012

[17] -
[http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/GetToolsInstall/GetToolsandInstall.html], 25.03.2012
[18] -
[http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html], 25.03.2012
[19] - [https://developer.apple.com/technologies/ios/cocoa-touch.html] 25.03.2012
[20] – "iOS Programming: The Big Nerd Ranch Guide, 2/e, kindle edition," page 291
[21] - [https://developer.apple.com/technologies/ios/data-management.html], 25.03.2012
[22] -
[http://www.ece.gatech.edu/academic/courses/ece4007/11spring/ECE4007L05/kj8/iOS SDK_TRP.pdf] 25.03.2012
[23] – Mark, D. LaMarche, J: "Beginning iPhone Development. Exploring the iPhone SDK," page 429-430
[24] - Dalrymple, M, Knaster, S: "Learn Objective-C on the Mac"
[25] -
https://developer.apple.com/library/ios/#releasenotes/General/WhatsNewIniPhoneOS/Articles/iOS5.html 06.04
[26] - Lott, J, Patterson, D: "Advanced ActionScript 3 with Design Patterns," page 46-47
[27] –
[http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/UIElementGuidelines/UIElementGuidelines.html] 26.04.2012
[28] – [https://github.com/jdg/MBProgressHUD] 28.04.2012
[29] – [http://code.google.com/p/skpsmtpmessage/wiki/GettingStarted] 28.04.2012
[30] – [https://github.com/davedelong/CHCSVParser] 28.04.2012
[31] –
[http://developer.apple.com/library/ios/#samplecode/Breadcrumb/Introduction/Intro.html#//apple_ref/doc/uid/DTS40010048] 28.04.2012
[32] – [http://www.apple.com/ios/features.html] 28.04.2012
[33] – [https://testflightapp.com/dashboard/] 30.04.2012

Appendix A

The survey for testers

The following questionnaire will help to study and analyze the Vehicle Mileage Log application made for my bachelor thesis. The survey consists of different tasks and questions about how difficult the tasks were to complete. If possible, please answer to all the questions. Thank you!

1. Add a new trip manually with yesterday's date.

1.1 * How easy it was adding a new trip manually?

- Very easy
- Easy
- Medium
- Hard
- Very hard
- Could not accomplish this task

1.2 Remarks

2. Change a made trip from private to business (or vice versa)

2.1 How easy it was to change a trip from private to business?

- Very easy
- Easy
- Medium
- Hard
- Very hard
- Could not accomplish this task

2.2 If changing the trip from private to business seemed hard, how would you change it?

2.3 Remarks

3. If possible, use GPS tracking for a new trip

3.1 How easy it was to end and save a trip?

- Very easy
- Easy
- Medium
- Hard
- Very hard
- Could not accomplish this task

3.2 What are the positive and negative sides of tracking a trip with GPS?

3.3 What would you change in tracking a trip with GPS?

3.4 Remarks

4. Add some made trip to favorites

4.1 * How easy it was to add a made trip to favorites?

- Very easy
- Easy
- Medium
- Hard
- Very hard
- Could not accomplish this task

4.2 What would you change in adding a trip to favorites?

4.3 How informative do you think the history overview of made trips is? What kind of information is missing and what is not needed?

4.4 Remarks

5. Send a report of made trips to your email

5.1 How easy was to send a report to email?

- Very easy
- Easy
- Medium
- Hard
- Very hard
- Could not accomplish this task

5.2 What would you change in sending a report to email?

5.3 Remarks

6. General evaluation of the application

6.1 How would you evaluate the following criteria for this application?

- **Intuitiveness – on a scale from 0 to 10**
- **User friendliness - on a scale from 0 to 10**
- **Easy to learn - on a scale from 0 to 10**
- **Easy to start using - on a scale from 0 to 10**

6.2 What are the good sides of this application?

6.3 What are the bad sides of this application?

6.4 What would you change in the application?

6.5 What kind of bugs or faults appeared in using the application?

Appendix B

The source code for the Vehicle Mileage log application is available on the CD attached to the back cover of the current thesis.