TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Software Science

Roland Talvar 112930IABB

# DISC GOLF IOS APP: ANALYSIS, DESIGN AND DEVELOPMENT

Bachelor's thesis

Supervisor: Enn Õunapuu
PhD

Tallinn 2017

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Roland Talvar 112930IABB

# DISCGOLF IOS RAKENDUS: ANALÜÜS, DISAIN JA ARENDUS

Bakalaureusetöö

Juhendaja: Enn Õunapuu
PhD

Tallinn 2017

# Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Roland Talvar

14.05.2017

# Abstract

Amateur disc golf players playing on Estonian disc golf courses don't have a good iOS application for accompanying the game. The purpose of this thesis is to build an iOS disc golf application for that.

The intended application can be used for locating courses, looking course details with current weather and course map, navigating to the course, keeping track of scores, in-course navigation, looking completed games and statistics. The scorekeeping supports multiple players with real-time synchronization between devices.

The thesis gives an overview of user research, existing solutions, application's analysis, design, development, and distribution.

As a result of this thesis, the iOS disc golf application is created, featuring all the required functionality. The application will be available on the iOS App Store for free.

This thesis is written in English and is 58 pages long, including 8 chapters, 26 figures and 2 tables.

# Annotatsioon

Discgolf iOS rakendus: analüüs, disain ja arendus

Eesti radadel mängivatel discgolfi harrastajatel ei ole iOS rakendust, mis oleks mängus heaks kaaslaseks. Käesoleva bakalaureusetöö eesmärgiks on luua iOS rakendus, mis on selleks mõeldud.

Planeeritud rakenduse funktsionaalsuse hulka kuulub: discgolfi parkide avastamine, parkide detailinfo kuvamine koos praeguse ilma ja pargikaardiga, parki navigeerimine, mängutulemuste salvestamine, pargisisene navigeerimine, tehtud mängude vaatamine ja statistika.

Rakendus toetab samas mängus olevate mängijate vahel mängutulemuste reaalajas sünkroniseerimist. Rakendust on lihtne kasutada, vastab modernse iOS rakenduse ootustele ning adapteerub erinevate ekraanisuurustega.

Töö annab ülevaate kasutajauuringust, olemasolevatest lahendustest ja nende võrdlusest. Töös tehakse rakenduse arendamiseks vajalik kontseptuaalne ja tehniline analüüs koos *UML* diagrammidega. Töö sisaldab ülevaadet kasutajaliidese arhitektuurist, Apple'i disainijuhistest ning mõningaid näiteid rakenduse disainimisest. Töös kirjeldatakse arenduseks kasutatavat programmeerimiskeelt, tööriistu, tarkvara disaini mustreid ja kolmanda osapoole teekide kasutamist. Samuti käsitletakse pideva integratsiooni protsessi ja rakenduse turustamist.

Töö tulemusena on loodud iOS rakendus, mis sisaldab planeeritud funktsionaalsust. Rakendus on kõigile tasuta saadaval iOS platvormi ametlikust rakenduste poest (*App Store*).

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 58 leheküljel, 8 peatükki, 26 joonist, 2 tabelit.

# List of abbreviations and terms

| | |
|---|---|
| App ID | String identifier to uniquely identify apps on Apple platforms |
| App Store | Platform for digital distribution of iOS apps maintained by Apple |
| Beta App Review | App review conducted by Apple which is needed for distributing apps to external beta testers through TestFlight |
| Developer Center | The environment for developers who develop on Apple platforms. It includes managing membership, certificates, IDs, and profiles; documentation, downloads, forums, bug reporter, news and updates. |
| Disc golf | Sports game similar to regular golf, but played with flying discs |
| Framework | "Framework is a bundle (a structured directory) that contains a dynamic shared library along with associated resources, such as nib files, image files, and header files" [1] |
| Git | Free and open-source distributed version control system for tracking changes in computer files |
| HIG | Human Interface Guidelines – Apple's document for design principles, rules and guidelines for designing iOS apps |
| iOS | Apple's mobile operating system used on iPhones, iPads, iPods and Apple TVs |
| iTunes Connect | "Suite of web-based tools created for developers to submit and manage their apps for sale in the App Store or the Mac App Store" [2] |
| JSON | JavaScript Object Notation – Human-readable lightweight data-interchange format derived from JavaScript |
| Library | Package of reusable pre-written code and data which can be used to achieve specific functionality |
| NIB | NeXTSTEP Interface Builder – File that describes user interface elements in iOS development |
| NoSQL | Non-relational database |
| Points | Logical unit for measuring user interface locations. 1 point equals to different number of pixels depending on the screen density |
| Provisioning profile | "A Provisioning Profile is a collection of digital entities that uniquely ties developers and devices to an authorized iOS |

| | |
|---|---|
| | Development Team and enables a device to be used for testing" [3] |
| Source code repository | Central place where source code is stored, usually accessible over the internet and used with version control system |
| Storyboard | "Visual representation of the user interface of an iOS application, showing screens of content and the connections between those screens" [4] |
| Struct | Value type general-purpose construct of code which encapsulates related properties and behaviors |
| Tee | In disc golf, an initial throwing place where game begins on each hole |
| TestFlight | Apple's service which provides distributing test versions of iOS applications |
| UI | User Interface – Part of the application that handles human-device interactions |
| URL | Uniform Resource Locator – Reference to a resource which is accessible on a computer network |
| UX | User Experience Design – Field which handles the process how a person interacts with the system |
| XML | eXtensible Markup Language – Human-readable markup language standard which is used to store and transport data |

# Table of Contents


**1 Introduction ..... 13**
**1.1 Disc golf ..... 13**
**1.2 Problems ..... 13**
**1.3 Goals ..... 13**
**1.4 Structure ..... 14**
**1.5 Methodology ..... 14**

**2 Market Research ..... 15**
**2.1 User research ..... 15**
**2.2 Existing solutions ..... 15**

**3 Analysis ..... 17**
**3.1 Conceptual analysis ..... 17**
3.1.1 Goals ..... 17
3.1.2 Statements ..... 17
3.1.3 Main objects ..... 20
3.1.4 Conceptual entity model ..... 21
3.1.5 Actors ..... 21
3.1.6 Use cases conceptual model ..... 21
3.1.7 Main events ..... 22
3.1.8 Main processes ..... 23
3.1.9 Information requirements ..... 23
3.1.10 Locations ..... 25
**3.2 Technical analysis ..... 25**
3.2.1 Data model ..... 25
3.2.2 Database structure ..... 26
**3.3 Other Requirements ..... 28**
3.3.1 Other Functional requirements ..... 28
3.3.2 Other Non-functional requirements ..... 29

**4 Design ..... 31**
**4.1 UI Architecture ..... 31**
4.1.1 Storyboards ..... 31
4.1.2 Implemented disc golf app storyboards ..... 31
4.1.3 Alternatives to storyboards ..... 31
**4.2 iOS Human Interface Guidelines ..... 33**
4.2.1 Introduction ..... 33
4.2.2 Examples of used guidelines ..... 33
**4.3 Managing keyboard ..... 34**
4.3.1 Text field positioning ..... 35
4.3.2 Text field rearrangements ..... 35
**4.4 Custom disc golf course map overlays ..... 36**

**5 Development ..... 37**
**5.1 Swift programming language ..... 37**
5.1.1 Introduction ..... 37

5.1.2 Functional programming ..... 37
**5.2 Xcode ..... 38**
**5.3 Software patterns ..... 38**
5.3.1 Model-View-Controller ..... 38
5.3.2 Delegation Pattern ..... 39
5.3.3 Composite validator pattern ..... 39
**5.4 Dependency management ..... 40**
5.4.1 Introduction ..... 40
5.4.2 Cocoapods ..... 40
5.4.3 Used 3rd party libraries ..... 40
**5.5 Real-time database ..... 40**
**6 Distribution ..... 42**
**6.1 Continuous Integration process ..... 42**
**6.2 Developer Center, iTunes Connect and App Store ..... 43**
**7 Results ..... 44**
**7.1 Future improvements ..... 44**
**8 Summary ..... 45**
**References ..... 46**
**Appendix 1 – Implemented storyboards ..... 48**
**Appendix 2 – Activity diagrams of main processes ..... 54**
**Appendix 3 – Source code repository ..... 59**

# List of figures

Figure 1. Conceptual entity diagram. 21
Figure 2. Conceptual use case diagram. 22
Figure 3. Data model. 26
Figure 4. Database structure example. 27
Figure 5. UI control 44x44 size example. 33
Figure 6. Reasonable default value example 1: Nearest disc golf course. 34
Figure 7. Reasonable default value example 2: Current user. 34
Figure 8. Reasonable default value example 3: Default score as a hole par. 34
Figure 9. Text field positioning without and with keyboard. 35
Figure 10. Text field rearrangement without keyboard. 35
Figure 11. Disc golf course map example. 36
Figure 12. Composite validator pattern. 39
Figure 13. Continuous integration process diagram. 42
Figure 14. Storyboard: Base with tab view controller. 48
Figure 15. Storyboard: Login. 49
Figure 16. Storyboard: Courses. 50
Figure 17. Storyboard: New course. 50
Figure 18. Storyboard: Game. 51
Figure 19. Storyboard: Play. 52
Figure 20. Storyboard: Profile. 53
Figure 21. Activity diagram: Creating new account. 54
Figure 22. Activity diagram: Signing in. 55
Figure 23. Activity diagram: Adding new disc golf course. 56
Figure 24. Activity diagram: Getting to the disc golf course. 57
Figure 25. Activity diagram: Starting new game. 58
Figure 26. Activity diagram: Playing the game. 58

# List of tables

Table 1. Disc golf applications and features. ..... 16
Table 2. Events according to use cases. ..... 22

# 1 Introduction

The motivation for this project grew out of a personal need while playing disc golf with a group of fellow disc golf fans. Being a huge fan of native mobile apps, especially iOS ones, I was trying to find a good complementary app which would enrich our game. Unfortunately, the options fell short of expectations. As a result, a decision was made – a new app must be born.

## 1.1 Disc golf

Disc golf is a sports game similar to a regular golf. But instead of clubs and balls, bare hands and flying discs are used.

Disc golf is played on a disc golf course. The course consists of holes. Hole start with a tee area and end with a chained basket. Players start the game with the first throw from the tee and continue where their throw landed. The aim is to get the disc to the basket with the fewest number of throws.

Disc golf popularity in Estonia has seen an explosive growth in the last years. There is currently approximately 100 [5] disc golf courses in Estonia and over 10 000 [6] estimated players.

## 1.2 Problems

There are no apps which are free, have course data prefilled and are easy to use. Also, none of the current solutions have scorekeeping with real-time synchronization between devices nor in-course navigation. Most of the apps have a user interface which doesn't meet the expectations of a modern iOS app.

## 1.3 Goals

The goal of this thesis is to build iOS disc golf application which is targeted for amateur disc golf players playing on courses in Estonia. The app will be downloadable from App

Store for free. Players should be able to use the application for discovering courses, getting course detail information, navigating to courses, tracking scores with real-time synchronization between devices, navigating between holes, looking completed games and statistics. The design should be adaptive for different screen sizes starting from 4" and conform to modern iOS user interface standards.

## 1.4 Structure

The 2nd chapter covers market research with user research and an overview of existing solutions. In the 3rd chapter, a conceptual and technical analysis is done for the app. The 4th chapter gives an overview of designing the app. The 5th chapter explains some of the development process, tools, and methodologies. The 6th chapter describes distribution processes. The results are analyzed in the 7th chapter along with possible future improvements.

## 1.5 Methodology

The application is developed in Xcode integrated development environment (IDE) using Swift 3 programming language. App uses Firebase cloud-based services including Realtime Database, Cloud Storage, Authentication, Cloud Functions and Cloud Messaging. 3rd party libraries are included in the app using Cocoapods dependency manager.

Application is designed using Apple's constraint-based layout system called Auto Layout. UI structure and navigation is laid out using storyboards.

For analysis, UML diagrams were made using VisualDesigner application.

Source code is uploaded to GitHub public repository. Every code push makes new version available for testers with continuous integration process set up using Nevercode, iTunes Connect and TestFlight.

# 2 Market Research

## 2.1 User research

In order to understand what the application should be like, a user research with a target group of amateur disc golf players was needed.

Since the questions that were needed to ask were “why” and “how”, not “how many” and “how much”, a qualitative (direct) research was conducted instead of a quantitative (indirect) one.

The research was done using both attitudinal and behavioural methods. It included interviews, focus groups, ethnographic field studies and concept testing with already existing solutions.

The research brought out some key points about what this application should be like:

- Free
- Easy to use
- Capability to include data about disc golf courses
- Easy to start navigation to the courses
- In-course navigation
- Real-time scorekeeping with synchronization between different devices
- Some sort of statistics: personal course records, completed games etc.

## 2.2 Existing solutions

There are several disc golf applications available on the iOS App Store, but none of them meet the requirements which were the outcome of the user research.

Here is a comparison chart of existing apps and user requirements:

Table 1. Disc golf applications and features.

| App name | Free | Easy to use | Course data | Navigate to course | In-course navigation | Realtime score keeping | Statistics |
|---|---|---|---|---|---|---|---|
| Disc Golf Course Review | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ |
| Disc Golf Course Locator | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Disc Golf Pro | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Disc Golf Tracker | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Udisc+ Disc Golf | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ |
| Frisbee Frolf | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ |
| Disctime | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Score Card for Disc Golf | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| iDisc Golf Scorecard | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Under Par Scorecard | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Longshot - PDGA Disc Golf | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Upsi Disc Golf | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ |
| Golf & Discgolf scorecard | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Disc Golf - PDGA | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ |

# 3 Analysis

## 3.1 Conceptual analysis

### 3.1.1 Goals

The following goals are set for the disc golf app:

- Users can sign in with email/password or with Facebook.
- Users can discover/search disc golf courses and navigate to them.
- Users can see detail information about disc golf courses including weather information and course map.
- Users can do the scorekeeping with multiple devices and real-time synchronization while playing disc golf.
- The app helps users to navigate between holes in course.
- Users can see their profile, completed games and statistics.

### 3.1.2 Statements

The following statements are describing the disc golf app:

- Disc golf is played in disc golf course.
- Disc golf is played by disc golf players.
- Disc golf course has coordinates.
- Disc golf course has a name.
- Disc golf course has holes.

- The hole has hole number.
- The hole has tee area.

- Tee has coordinates.
- The hole has a basket.
- The basket has coordinates.
- The hole has par.
- The course has par which is equal to the sum of holes' pars.

- User has a name.
- User has an email.
- User needs to authenticate to be able to use the app.
- User signs in with Facebook.
- User's profile picture is downloaded from Facebook.
- User registers with email and password.
- User uploads profile picture while registering.
- User signs in with email and password.
- User logs out from the app.

- Player is user
- Administrator is user
- User is player and/or administrator
- User can be marked as an administrator by the creator of the app

- App has a list of disc golf courses.
- User searches/discovers disc golf courses on map.
- User searches/discovers disc golf courses in the list.
- User looks disc golf course detail information.
- User looks current weather of disc golf course.
- User looks disc golf course detail map (holes: tees and baskets).
- User uses navigation to reach disc golf course.
- Administrator adds new disc golf course.
- Administrator enters name, number of holes and total par to the disc golf course.
- Administrator marks current location as the coordinates of the course.
- Administrator enters holes (with hole number and par) to the disc golf course.
- Administrator marks current location as tee coordinates of the hole.
- Administrator marks current location as basket coordinates of the hole.

- User starts new game.
- User who is added to the game is a player.
- Game has a start time.
- Game has an end time.
- Player chooses a disc golf course where the game takes place.
- App shows a list of other potential players.
- Player can add other players to the game.

- Player gets a push notification if he/she is added to a game.
- By default, app sets the hole par as the player's score.

*While game is ongoing, ...*

- …app shows scorecard with course name and number of holes.
- …app shows current hole information: hole number, par.
- …app shows list of players and their score of current hole, total score and difference from the course par.
- …player updates his/her score.
- …if player's score is updated, it is synchronized to other players' apps real-time.
- …player uses in-course navigation to navigate between holes.

- App has a profile view.
- Profile view includes user's profile picture, name and email.
- User looks personal records of disc golf courses under profile view.
- User looks completed games under profile view.

### 3.1.3 Main objects

Main objects of the app are:

- Disc golf course
- Hole of the course
- User
- Game

### 3.1.4 Conceptual entity model

A conceptual model of main entities in the app:

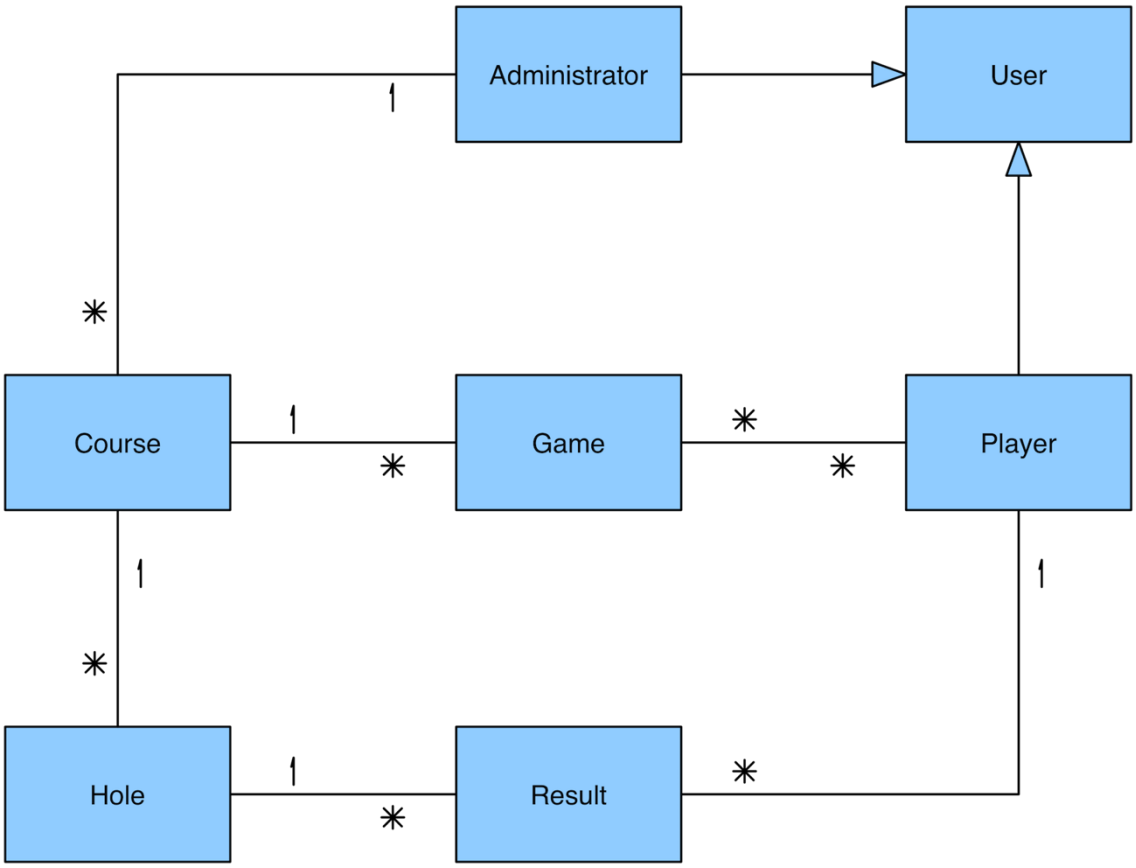


Figure 1. Conceptual entity diagram.

### 3.1.5 Actors

Actors of the app are:

- Player
- Administrator

### 3.1.6 Use cases conceptual model

A conceptual model of main use cases in the app:

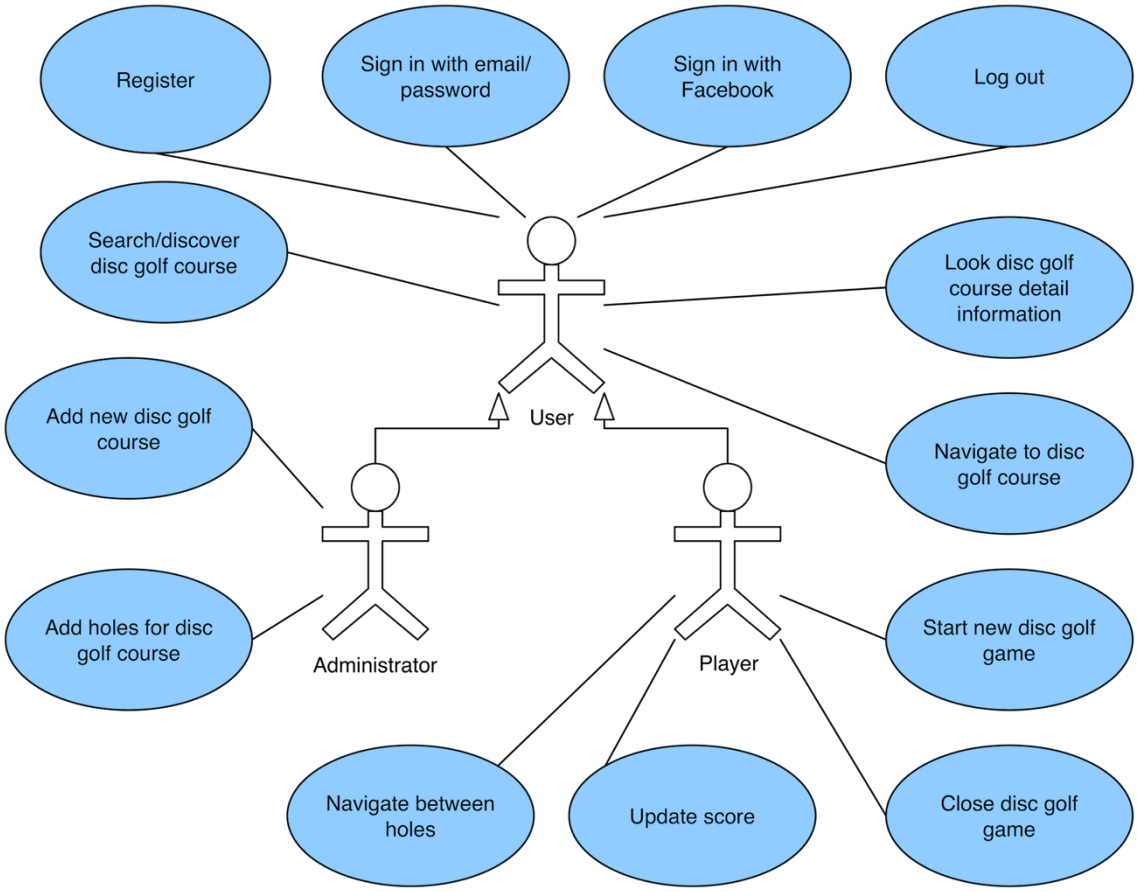


Figure 2. Conceptual use case diagram.

### 3.1.7 Main events

In the following table is a list of main events according to use cases:

Table 2. Events according to use cases.

| Event | Use case |
|---|---|
| User installs the app first time and doesn't wish to sign in with Facebook | Register |
| User installs the app and has previously registered with email/password | Sign in with email/password |
| User installs the app first time and wishes to sign in with Facebook or has previously signed in with Facebook | Sign in with Facebook |
| User wishes to close the authentication session | Log out |
| User wants to discover new disc golf courses, find the most nearby course or see information about specific course | Search/discover disc golf course |
| User wants to see detail information about specific course. | Look disc golf course detail information |

| Event | Use case |
|---|---|
| User wants to use a navigation app to navigate to the course – such as Google Maps or Apple Maps | Navigate to disc golf course |
| Administrator discovers that a new disc golf course is created or existing one is not yet in the app database | Add new disc golf course |
| Administrator wishes that users can see holes' information and use in-course navigation | Add holes for disc golf course |
| User starts playing disc golf | Start new disc golf game |
| User has finished current hole and updated scores | Navigate between holes |
| User has done a throw or finished hole (if user wishes to update score once per hole) | Update score |
| User stops playing disc golf | Close disc golf game |

### 3.1.8 Main processes

Main processes on the app are:

- Creating new account
- Signing in
- Adding new disc golf course
- Getting to the disc golf course
- Starting a new game
- Playing the game

Conceptual activity diagrams of these main processes are added to Appendix 2.

### 3.1.9 Information requirements

The app needs following information:

1. List of disc golf courses and their detail information:

    a. Name
    b. Coordinates
    c. Number of holes
    d. Course par
    e. Holes included
    f. Current weather
    g. Website
    h. In-course navigation availability

2. List of holes in the course and their detail information:
    a. Hole number
    b. Tee coordinates
    c. Basket coordinates
    d. Par

3. List of games and their detail information:
    a. Course
    b. Players
    c. Results
    d. Start time
    e. End time
    f. If game is completed

4. List of users and their detail information:

a. Name

b. Email

c. Profile picture

d. List of completed games

e. List of ongoing games

5. Authentication and authorization information

6. Map data

### 3.1.10 Locations

The app is associated with following locations:

- App is downloadable from App Store (iOS) which is maintained by Apple.
- Users can use the app anywhere in the world using their iOS devices.
- If user has authenticated and app has downloaded necessary data then user can use the app in locations without network connection.
- While users are playing a disc golf game they are located on a disc golf course.
- In able for administrator to add new course, one must be located on the course because coordinates are marked using current device location.
- App's data is cached on device and stored in Firebase cloud service which is owned by Google.
- If user signs in with Facebook then user's profile picture is hosted by Facebook.

## 3.2 Technical analysis

### 3.2.1 Data model

The main model classes in the app:


Figure 3. Data model.

### 3.2.2 Database structure

The database used in the project is Firebase database. This is a NoSQL database and is essentially a large JSON document. Therefore, it doesn't have a database schema like a SQL database would.

Structuring the data in Firebase NoSQL database is quite different from the SQL database standpoint. There is no query language and any kind of joins are not possible.

Here is an example snippet from the database:

```
{
  "courses" : {
    "-KjyDrADE1uynojEa83e" : {
      "coordinates" : "58.9806853747545, 26.255229916448",
      "hasInCourseNavigation" : true,
      "holes" : {
          "hole1" : {
            "basketCoordinates" : "58.9800036745695, 26.2546673231071",
            "nr" : 1,
            "par" : 3,
            "teeCoordinates" : "58.9806281682653, 26.2550572492427"
          }
        },
      "name" : "Rakke DiscGolfPark",
      "nrOfHoles" : 9,
      "par" : 27,
      "website" : "https://www.discgolfirajad.ee/park/rakke-discgolfi-park/"
    }
  },
  "users" : {
    "Mspllwl3bjaAeX0M6HyG6BStkNA3" : {
      "completedGames" : {
        "-Kjv1bsiYYkHFjDgICpp" : 0,
        "-Kjv6vr-DYsenqH0s8O4" : 0
      },
      "email" : "talvar.roland@gmail.com",
      "name" : "Roland Talvar",
      "ongoingGames" : {
        "-Kk1MhhqSik4sl6-PpeS" : 0
      },
      "pictureUrl" : "https://scontent.xx.fbcdn.net/v/t1.0-1/p100x100/119du"
    }
  }
}
```


Figure 4. Database structure example.

As seen from the example, the data is not completely normalized. This is deliberately so, because specific application information needs and network requests were considered while structuring the database. In other words, the database structure is designed based on app's data usage.

Another bizarre thing that can be seen in this example is "completedGames" value. This is actually an array of game id's and associated "0" values don't have a meaning. There are 2 reasons why it's done like this:

1. There is no native support for arrays in Firebase. Everything is a dictionary, because there can be unexpected results when multiple clients access same array at the same time [7].

2. Firebase will delete all keys of which value is *null* [8].

## 3.3 Other Requirements

### 3.3.1 Other Functional requirements

- App should validate data entered to the form fields and show informational feedback if the data is invalid.
- App should show appropriate data input type according to the type of field (for example field for integer should open numeric keyboard, field for email should open email keyboard etc.).
- App must change entered text in password fields to unreadable symbols.
- If there is another field in the form and user taps enter key on the keyboard, app should change the focus to the next field.
- If focus is on the last field in the form and user taps enter key, app should act like the button associated with the form was tapped.
- If back-end sends an error message back to the app, app should show the message to the user.
- App should show loading indicator during network requests.
- User must stay logged in if app is closed.
- App should use Google Maps for navigating to the course and if Google Maps is not installed then use Apple Maps as a secondary choice.
- Back-end could send push notification to player's device if this player is added to a new game by another player.
- App could have a possibility to add non-registered players to the game.
- App could have a possibility to play game in a disc golf course which is not in the database.
- Currently, the app won't have to include all disc golf courses in Estonia, but could include them in the future.

- App must include at least 5 disc golf courses on launch.
- App could be accompanied by an Apple Watch app in the near future.
- App could include a possibility to export game results (for example, PDF or PNG format) and share them using native iOS share sheet.

### 3.3.2 Other Non-functional requirements

- Back-end system (database, storage, authentication and push notifications) must have cross-platform support to be able to use with possible future front-end applications on other platforms (for example: Android, web application).
- App must use real-time database in order to keep game synchronized between players.
- App must be developed using Swift programming language which is of version at least 3.0.
- Graphical user interface should be in accordance to the Apple User Interface Guidelines.
- Graphical user interface should be made in accordance to modern iOS design best practices.
- Graphical user interface must be responsive and support all iOS device display sizes from 4.0".
- App must not save user credentials. App must use safe industry standard cloud service for user authentication.
- App should cache information from cloud database to improve loading times, use less cellular data and offer offline capabilities.
- App should validate form information.
- App's orientation must be locked to be portrait only.
- App must use composed validation pattern to do field validations.

- 3rd party libraries must be installed using dependency manager, preferably Cocoapods.
- App should use Swift's functional programming features where possible.
- App must be built using storyboards which are logically divided with storyboard references.
- App layout must be built using Apple's "Auto Layout".
- String identifiers should be declared in global nested *struc*t constants.

# 4 Design

## 4.1 UI Architecture

### 4.1.1 Storyboards

The user interface of the app is laid out using storyboards, which is the Apple's recommended way to set up modern iOS application's visual representation. Using storyboards, it is easy to understand the view structure and application flow.

Storyboard is composed of scenes, which include a single view controller and its associated view. Scenes are connected with segues, which help to reduce boilerplate code for navigation and transitions.

Because storyboard is essentially a big non-human-readable (because of auto-generated id's) XML file, it can cause conflicts if several people are modifying it at the same time. Also, if the file is too big, it can become hard to navigate and slow down the IDE. Therefore, it is best practice to divide storyboard files into smaller ones and use storyboard references introduced in iOS 9.

### 4.1.2 Implemented disc golf app storyboards

Implemented storyboards are added to Appendix 1.

### 4.1.3 Alternatives to storyboards

Other options to build user interface are NIB files or code. NIB files represent a single view without segues and view controllers and code is self-explanatory. There is no universal best choice which tool to use for building the UI. It depends on the project type, size, and development team. It is reasonable to use all three to complement each other's strengths and weaknesses [9].

Storyboard pros:

- Easy to get started
- Helps to visualize and understand app's UI architecture and flow
- Fast prototyping

- Performance (view controllers instantiated when needed)
- Least amount of boilerplate code
- Apple's recommended way, therefore, it has a lot of examples and tutorials

Storyboards cons:

- Reusability
- Not good with dynamic layouts
- Possible merge conflicts
- Can slow down IDE and make navigating the views cumbersome

NIB pros:

- Reusability of a single view
- Visualization of a single view
- Less boilerplate code than "code" (but more than storyboard)

NIB cons:

- Doesn't visualize app flow
- Not good with very dynamic layouts (better than storyboard)
- Possible merge conflicts (but to a lesser extent compared to a storyboard)

Code pros:

- Reusability
- No merge conflicts
- Only option to use when building highly dynamic views

Code cons:

- No visualization at all, hard to prototype and understand app flow
- A lot of boilerplate code
- Hard to get started

In this project storyboards and code is used to build the user interface. NIB files are not used because there is no particular need for them in this case.

## 4.2 iOS Human Interface Guidelines

### 4.2.1 Introduction

Human Interface Guidelines (HIG) [10] is an Apple's official documentation about how the UI of iOS apps should be built. It has general design theories, philosophies, guidelines how app should act and strict rules about specific views.

### 4.2.2 Examples of used guidelines

- " **Create controls that measure at least 44 points x 44 points** so they can be accurately tapped with a finger" [11]

Most of the UI controls are at least 44 points wide and high. There are few exceptions when the size is not reasonable. In this example screenshot, text fields and important buttons follow the guidelines, but the "Forgot password?" button neglects it:

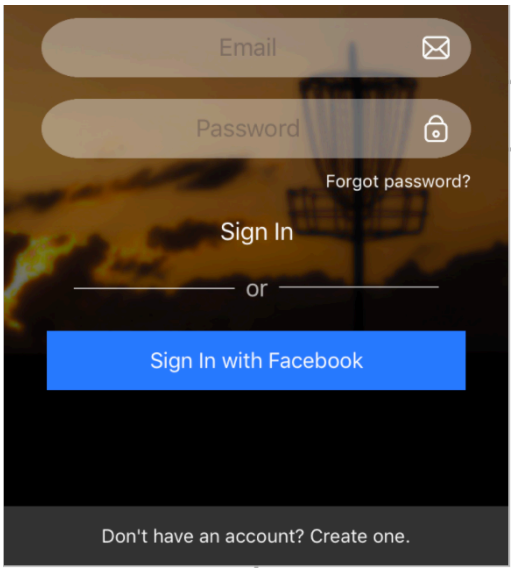


Figure 5. UI control 44x44 size example.

- **“Provide reasonable default values”** [12]

The app preselects values where it is reasonable. For example, if a user starts playing, it preselects nearest disc golf course, preselects current user as a player and sets hole par as a default value for the score:

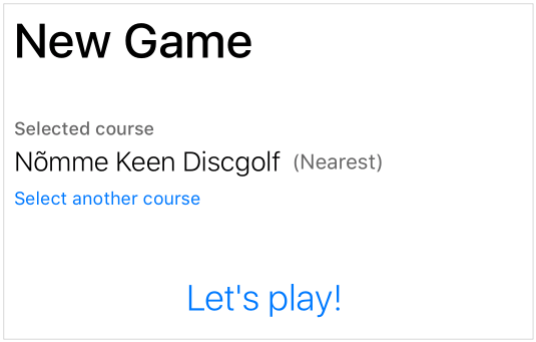


Figure 6. Reasonable default value example 1: Nearest disc golf course.

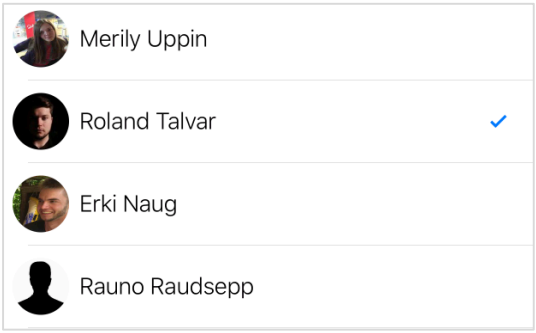


Figure 7. Reasonable default value example 2: Current user.

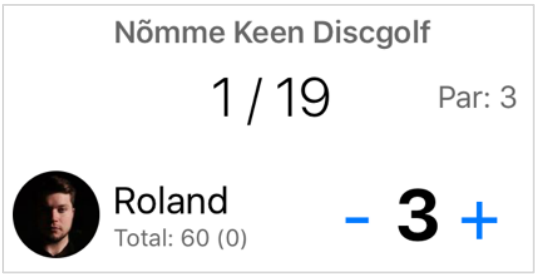


Figure 8. Reasonable default value example 3: Default score as a hole par.

## 4.3 Managing keyboard

When user touches text field, iOS displays a keyboard which hides portion of the screen. It is important to handle such cases so that keyboard doesn’t interrupt user’s forward interactions with the app (hide the text field). In this app, 2 different strategies are used to address this issue: text field positioning and text field rearrangement.

### 4.3.1 Text field positioning

If possible, text fields are positioned in a way that keyboard doesn't hide them:

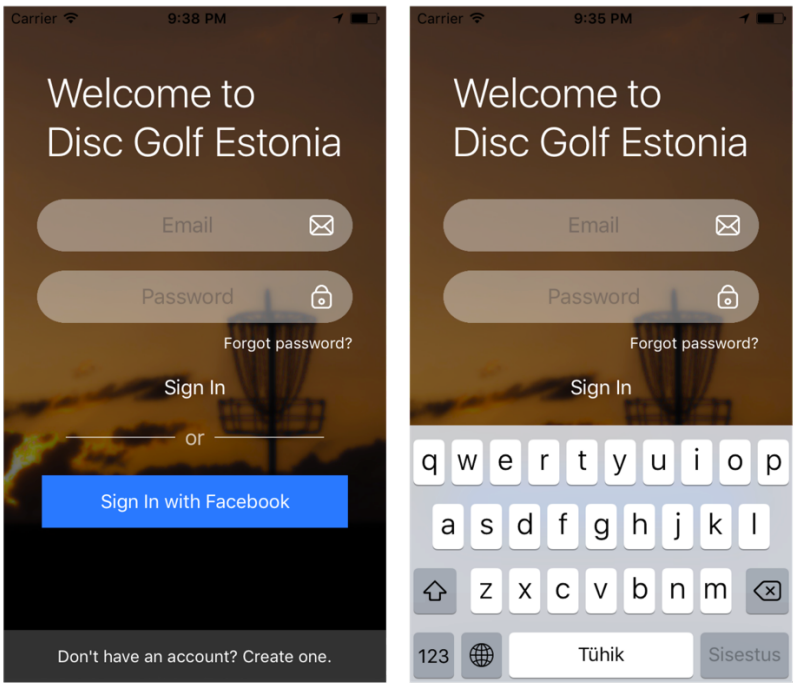


Figure 9. Text field positioning without and with keyboard.

### 4.3.2 Text field rearrangements

If there are too many text fields then text fields are rearranged on keyboard appear:

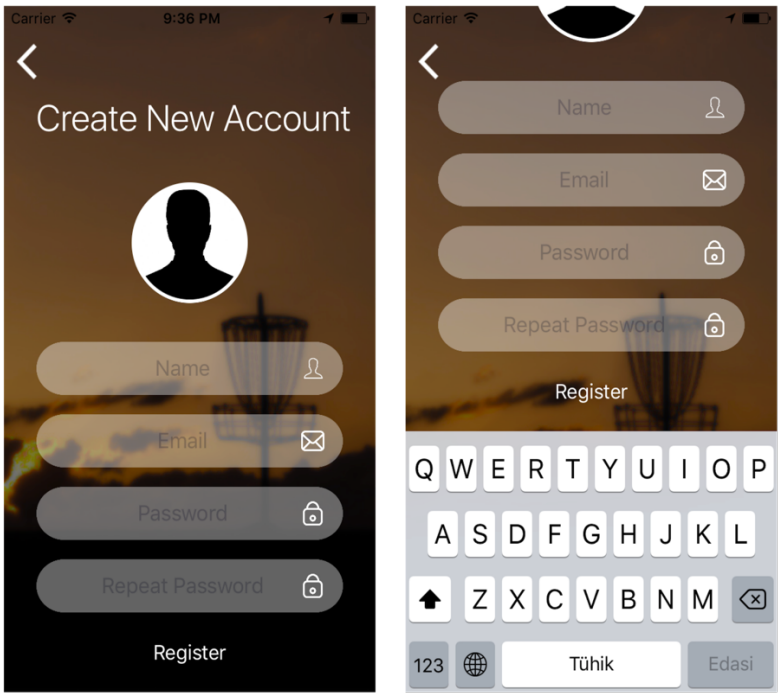


Figure 10. Text field rearrangement without keyboard.

## 4.4 Custom disc golf course map overlays

To better visualize disc golf courses on the map, custom map pins are added for tees and baskets. Tee pins are numbered with hole numbers. For better clarity, routes are drawn between tees and baskets.

Example screenshot of a disc golf course map:

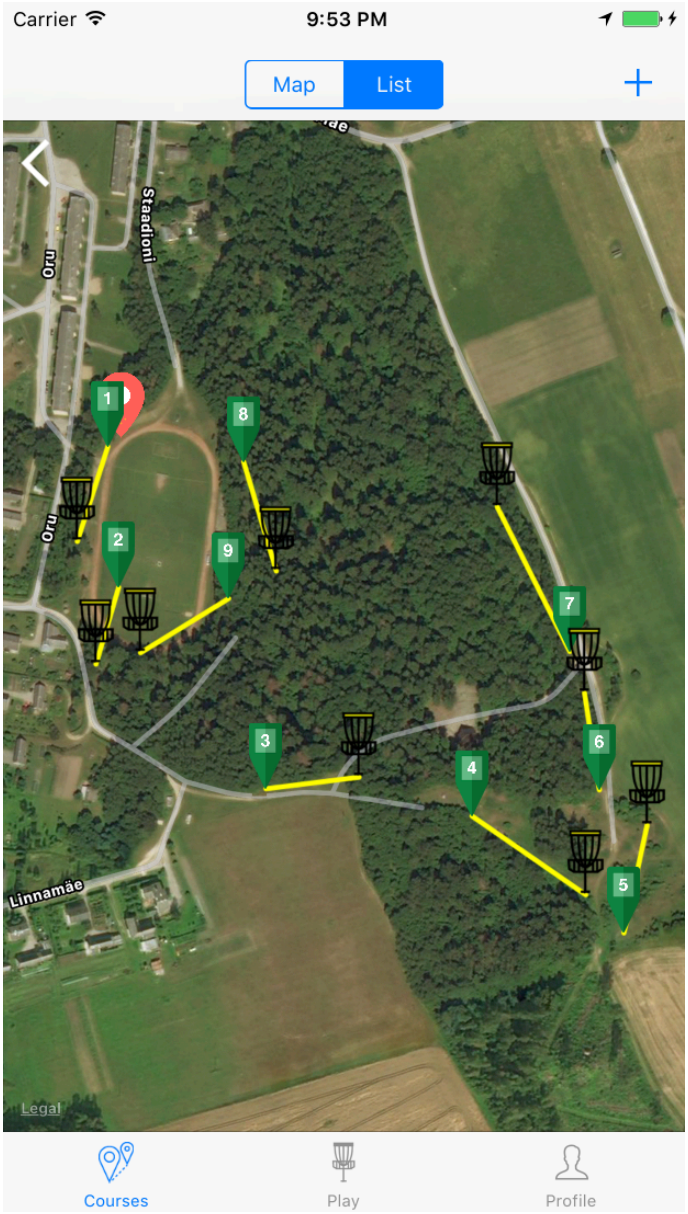


Figure 11. Disc golf course map example.

# 5 Development

## 5.1 Swift programming language

### 5.1.1 Introduction

Swift programming language was introduced at Apple's 2014 Worldwide Developer Conference as a replacement for Objective-C – the original iOS development language. From then it has been gaining popularity fast and according to the TIOBE Index, surpassed Objective-C in the beginning of 2016.

It is a general-purpose, multi-paradigm, compiled programming language which is intended to be safe, fast and expressive. "It is the first industrial-quality systems programming language that is as expressive and enjoyable as a scripting language" [13]. Swift encourages the use of value type and protocol-oriented programming. It was designed by Apple with Chris Lattner in the lead role. On December 3, 2015, Swift was open-sourced under Apache 2.0 license and anyone can contribute to the future development.

### 5.1.2 Functional programming

Swift has some influences of functional programming. It is not a purely functional programming language like Haskell, but it has a support for higher order functions – functions which take other functions as arguments and/or return other functions as a result. Here are some examples where functional programming was used in the project.

1. Converting dictionary of holes to an array of holes:

    a. Imperative:

```
var holesArray = [Hole]()

for hole in holes {
    holesArray.append(hole.value)
}

delegate?.didSaveHole(sender: self, holes: holesArray)
```

    b. Functional:

```
delegate?.didSaveHole(sender: self, holes: holes.map { $1 })
```

2. Calculating the sum of pars in one course:

    a. Imperative:

```
var totalParOfHoles = 0

for hole in holes {
    totalParOfHoles += hole.par
}
```

    b. Functional:

```
let totalParOfHoles = holes.reduce(0) {$0 + $1.par}
```

## 5.2 Xcode

Xcode is the only tool that an iOS developer needs to download. It is an integrated development environment (IDE) developed by Apple. Xcode contains source code editor, iOS SDK (Software Development Kit), Interface Builder, debugging tools, simulators and more. Since it is also the only IDE that has all the necessary tools, it is used for developing this project.

## 5.3 Software patterns

A good software developer uses best practices which are developed over time by the community. One of these best practices is using software patterns. “In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design [14]. By using software patterns, developers don’t have to re-invent the wheel. Patterns also allow better communication with a common vocabulary for well-known high-level ideas.

### 5.3.1 Model-View-Controller

MVC is a central design pattern in iOS applications and frameworks. It helps to make the code more reusable and contributes to applying the separation of concerns design principle. “The Model-View-Controller (MVC) design pattern assigns objects in an application one of three roles: model, view, or controller. The pattern defines not only the roles objects play in the application, it defines the way objects communicate with each other. Each of the three types of objects is separated from the others by abstract

boundaries and communicates with objects of the other types across those boundaries" [15].

### 5.3.2 Delegation Pattern

Delegation pattern is among the most common patterns in iOS development and. It is heavily used by Apple's frameworks. It greatly describes how Cocoa frameworks are built. "Delegation is a simple and powerful pattern in which one object in a program acts on behalf of, or in coordination with, another object. The delegating object keeps a reference to the other object—the delegate—and at the appropriate time sends a message to it. The message informs the delegate of an event that the delegating object is about to handle or has just handled" [16]. In Apple's UI frameworks, many UI elements use delegates to control their behavior.

### 5.3.3 Composite validator pattern

Composite validator pattern is a result of applying the composite pattern to input. "The composite pattern is a design pattern in which a group of objects can be treated the same way as a single object. The idea is to compose objects into tree structures to represent part-whole hierarchies" [17].

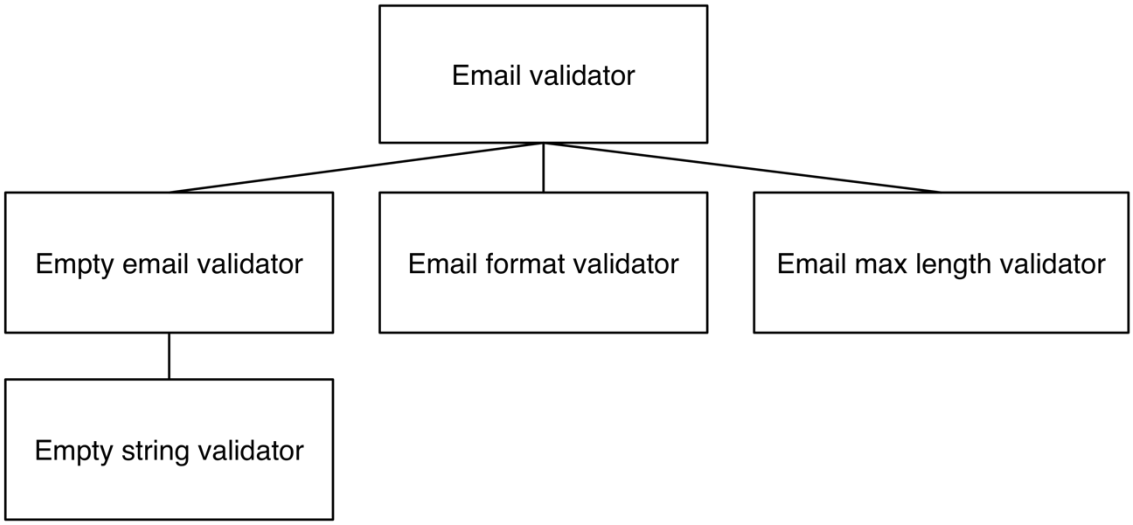


Figure 12. Composite validator pattern.

The composite validator pattern is used in this project to validate user input.

## 5.4 Dependency management

### 5.4.1 Introduction

iOS applications are usually built using 3rd party libraries. It is done, because most of the applications use some common functionality, like networking and social login. Libraries help developers to save time and in most cases, it is impossible to recreate the same functionality and performance in a reasonable amount of time.

Dependency managers make using 3rd party libraries simple. Instead of manually adding the code to the project, library name is written into dependencies file. It also updates the libraries and takes care of version conflicts.

### 5.4.2 Cocoapods

Cocoapods is the most popular application level dependency manager for Swift and Objective-C Cocoa projects. It has over 32 thousand libraries and is used in over 2.1 million apps [18]. Because of best support and most number of libraries, it is the default choice for iOS developers. That is why it is chosen to be used in this project as well.

### 5.4.3 Used 3rd party libraries

Here are some 3rd party libraries used in the project:

- FacebookLogin - Facebook authentication and profile information
- Firebase (/Core, /Auth, /Storage, Database) – Firebase cloud services
- Kingfisher – Downloading and caching images from the web
- Jupiter – Framework for Dark Sky weather API

## 5.5 Real-time database

One of the main features of the app is live scorekeeping synchronization between different instances of the app (different users with their devices). To avoid polling the server with network requests manually, cloud-hosted real-time database is used.

Currently, there are 2 commercial and several open-source solutions for cloud real-time database with support for iOS. Open-source solutions are either too restrictive or not

ready yet. Commercial options are Realm Mobile Platform and Firebase. In this application, Firebase is used for following reasons:

- More mature. As of this writing, Realm Mobile Platform is still in beta.
- Better infrastructure with analytics, database, storage, authentication, cloud messaging and cloud functions.
- Support for anonymous authentication.

# 6 Distribution

## 6.1 Continuous Integration process

Continuous integration helps to automate the process which starts with pushing code from developers' machine and ends with usable application in testers' devices.

The continuous integration process used while developing the app is following:

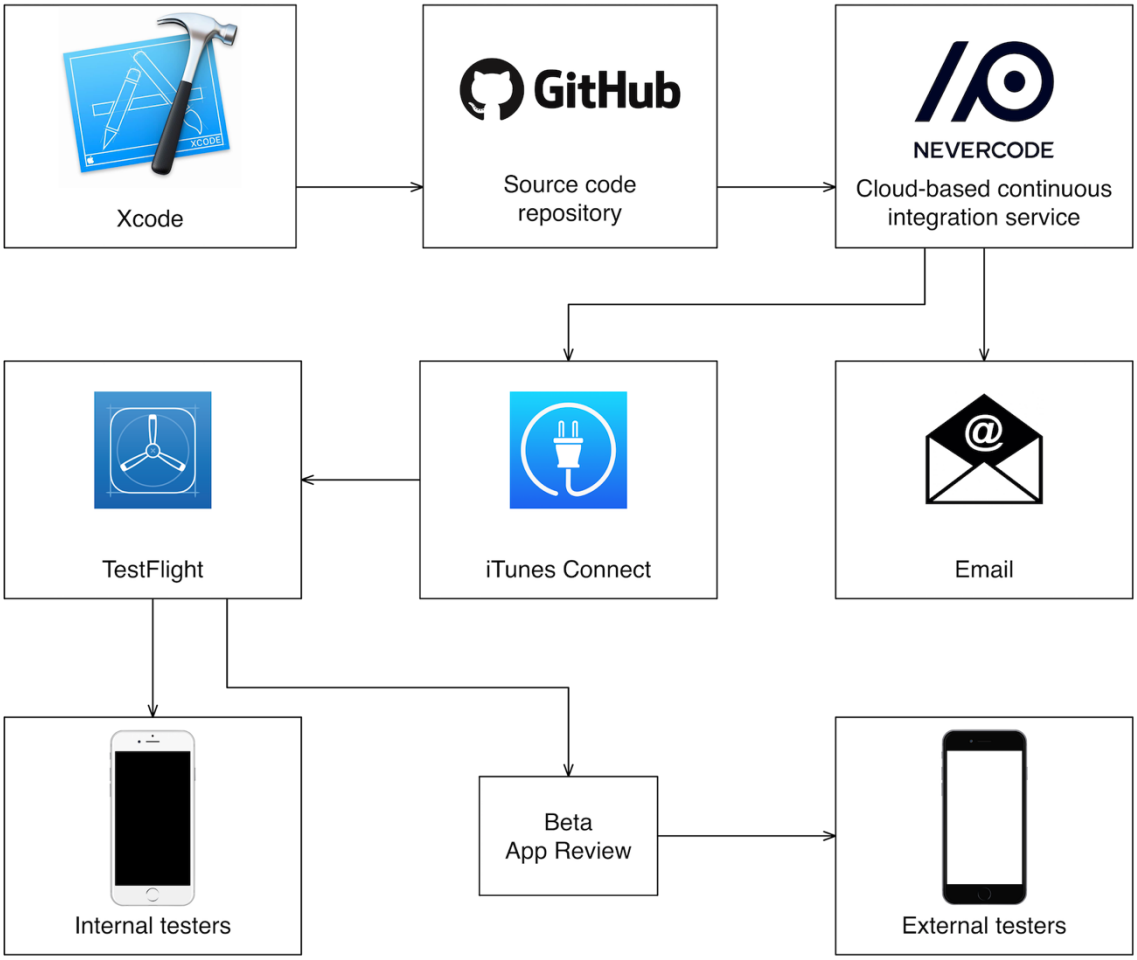


Figure 13. Continuous integration process diagram.

1) Code is committed and changes are pushed to GitHub source code repository using Git.

2) A post-receive git hook (script file) is triggered in GitHub which notifies Nevercode that new code is available.

3) Nevercode fetches the code from GitHub, signs it with provided Provisioning profile and certificate and builds it.

4) If build is successful, Nevercode publishes built .ipa binary file to iTunes Connect and sends notification email with build information.

5) iTunes Connect processes the build and makes it available to the TestFlight.

6) TestFlight makes new app version available to internal testers and sends them notification that new update is ready to install.

7) App can be submitted for a Beta App Review. If the review is successful, app can be made available to external testers also.

## 6.2 Developer Center, iTunes Connect and App Store

For distributing an app on the App Store, developers need to work with 3 environments: Developer Center, iTunes Connect and App Store. Before distribution developers need to enroll in Apple Developer Program for a 99 USD annual fee.

To distribute the app, it must be properly signed. “Code signing your app assures users that it is from a known source and the app hasn’t been modified since it was last signed” [19]. Signing needs a distribution provisioning profile and a certificate issued by Apple. Creating a provisioning profile needs App ID and this same certificate. All of this can be managed in Developer Center.

iTunes Connect is used to manage the app before submitting to the App Store. It processed the builds and notifies if there is an issue with your app. It is used to fill marketing information (videos and screenshots for app preview, description, keywords, marketing URL), support-, contact- and copyright information. It is also for managing app versions, availability and pricing. If everything is done and app has successfully passed App Review, it can be finally submitted to the App Store.

# 7 Results

As an outcome of this thesis, an iOS disc golf application was built. Application includes all required features including locating courses using map and list, getting course detail information with current weather and course map, navigating to courses, tracking scores with real-time synchronization between devices, navigating between holes and statistics. The app is easy to use, navigation and UX is familiar for an iOS user. Initial testing has shown that the features are useful and the app is a good companion for disc golf.

## 7.1 Future improvements

Currently the 2 biggest issues with the app are:

1. Course data – The app has all necessary capabilities for managing course data, but it doesn't include data for all the courses in Estonia. There is no central database yet for Estonian disc golf courses, which API could be used. Therefore, adding course data is currently handled manually. Also, if courses change, the course data has to be updated manually.

2. Branding – The app has too generic name, logo and visual design. It could use some branding which would include new name, new logo and recognizable visual design.

In addition to these, possible developments in the future are:

1. Apple Watch app – Players can update their score with tapping on the wrist after every throw. Since taking smartphone out of the pocket after every throw is cumbersome, players usually update their scores after hole is completed. But this leads to an increased cognitive load of memorizing throws and possible errors with remembering the correct score.

2. Exporting results to a PDF and/or PNG format, sharing with iOS native share sheet.

3. Disc Golf Metric integration. Players can upload their results.

# 8 Summary

The main goal of this bachelor thesis was to create easy to use iOS disc golf application for amateur disc golf players. The application was required to have functionality for, among other things, finding disc golf courses, looking detail information about courses, keeping scores with real-time synchronization between devices, showing personal course records and completed games.

In the thesis author described market research, analysis, design, development, and distribution of the application.

As a result of this thesis, the author successfully created an iOS application with features that were required.

The scope of this thesis didn't include having data about all the disc golf courses in Estonia. Having a central database with the data about all the courses is a big topic possible for another thesis.

The URL to source code repository of the app is added to Appendix 3.

The application is submitted for an App Store review. If the review is successfully passed, it is ready for public use.

# References


[1] Apple, Cocoa Core Competencies [WWW] https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Framework.html (19.04.2017)

[2] Apple, About iTunes Connect [WWW] https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Chapters/About.html (19.04.2017)

[3] Apple, App Distribution Quick Start [WWW] https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/CreatingYourTeamProvisioningProfile/CreatingYourTeamProvisioningProfile.html#//apple_ref/doc/uid/TP40013839-CH33-SW1 (19.04.2017)

[4] Apple, Cocoa Application Competencies for iOS [WWW] https://developer.apple.com/library/content/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html (19.04.2017)

[5] Discgolfi rajad [WWW] https://www.discgolfirajad.ee (19.04.2017)

[6] Sportland, Sportland Magazine [WWW] http://www.sportlandmagazine.com/est/disc-golf-uha-populaarsem-ala/ (19.04.2017)

[7] K. Richardson, Best Practices: Arrays in Firebase, Firebase [WWW] https://firebase.googleblog.com/2014/04/best-practices-arrays-in-firebase.html (8.04.2017)

[8] Firebase, Firebase Documentation - Saving Data [WWW] https://firebase.google.com/docs/database/admin/save-data (8.04.2017)

[9] A. Bello, iOS User Interfaces: Storyboards vs. NIBs vs. Custom Code [WWW] https://www.toptal.com/ios/ios-user-interfaces-storyboards-vs-nibs-vs-custom-code (9.04.2017)

[10] Apple, iOS Human Interface Guidelines [WWW] https://developer.apple.com/ios/human-interface-guidelines/ (10.04.2017)

[11] Apple, UI Design Do's and Don'ts [WWW] https://developer.apple.com/design/tips/ (10.04.2017)

[12] Apple, Data Entry [WWW] https://developer.apple.com/ios/human-interface-guidelines/interaction/data-entry/ (10.04.2017)

[13] Apple, About Swift [WWW] https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/ (10.04.2017)

[14] Wikibooks, Introduction to Software Engineering/Architecture/Design Patterns [WWW] https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Architecture/Design_Patterns (11.04.2017)

[15] Apple, Model-View-Controller [WWW] https://developer.apple.com/library/content/documentation/General/Conceptual/DevP

edia-CocoaCore/MVC.html (12.04.2017)

[16] Apple, Delegation [WWW] https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Delegation.html (12.04.2017)

[17] S. Robins, Composite Validators [WWW] http://hotcocoatouch.com/2016/11/16/composite-validators/ (12.04.2017)

[18] CocoaPods, CocoaPods [WWW] https://cocoapods.org (13.04.2017)

[19] Apple, Code Signing [WWW] https://developer.apple.com/support/code-signing/ (13.04.2017)

# Appendix 1 – Implemented storyboards

Base storyboard with tab bar controller:

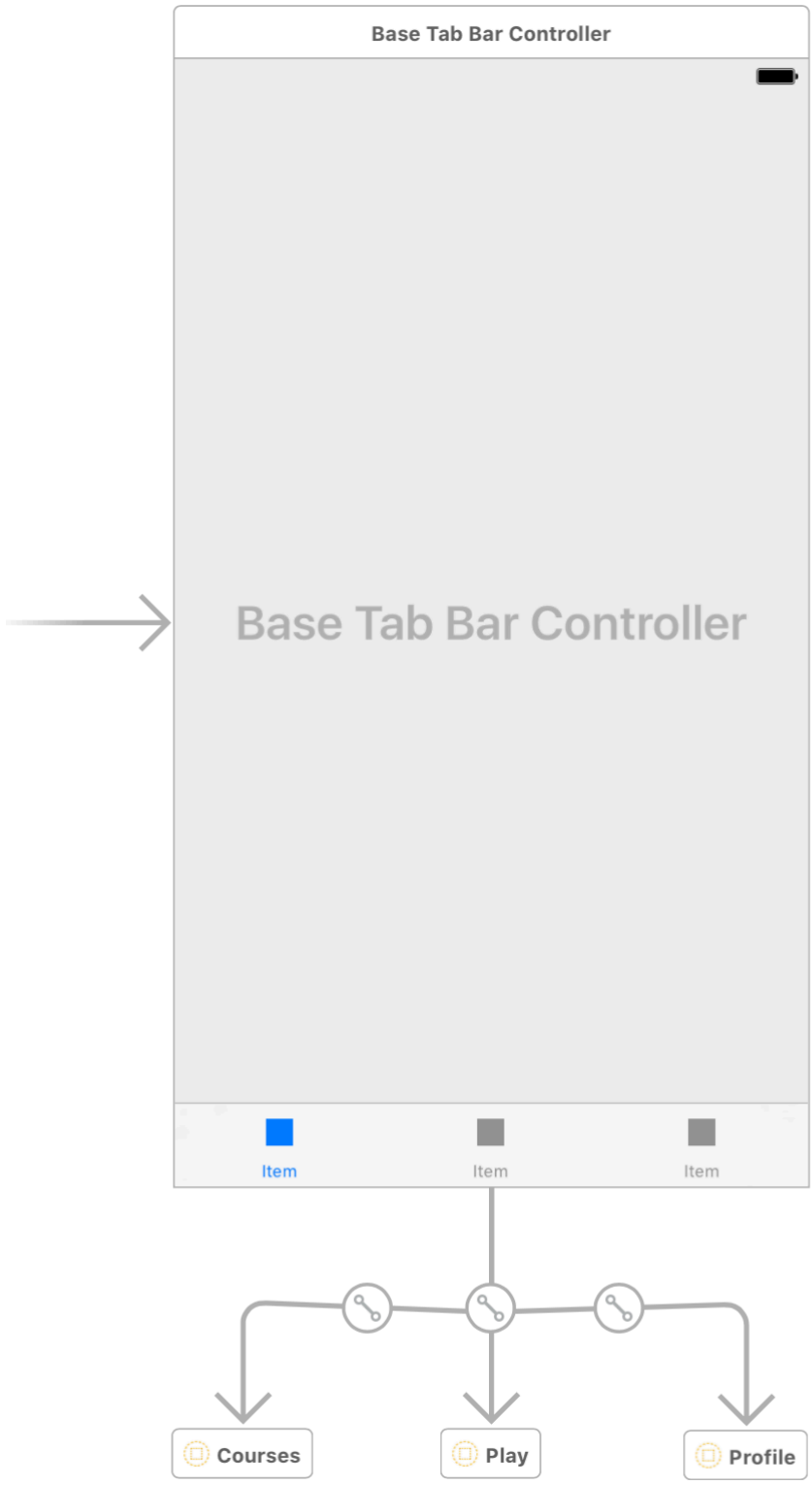


Figure 14. Storyboard: Base with tab view controller.

Login storyboard:

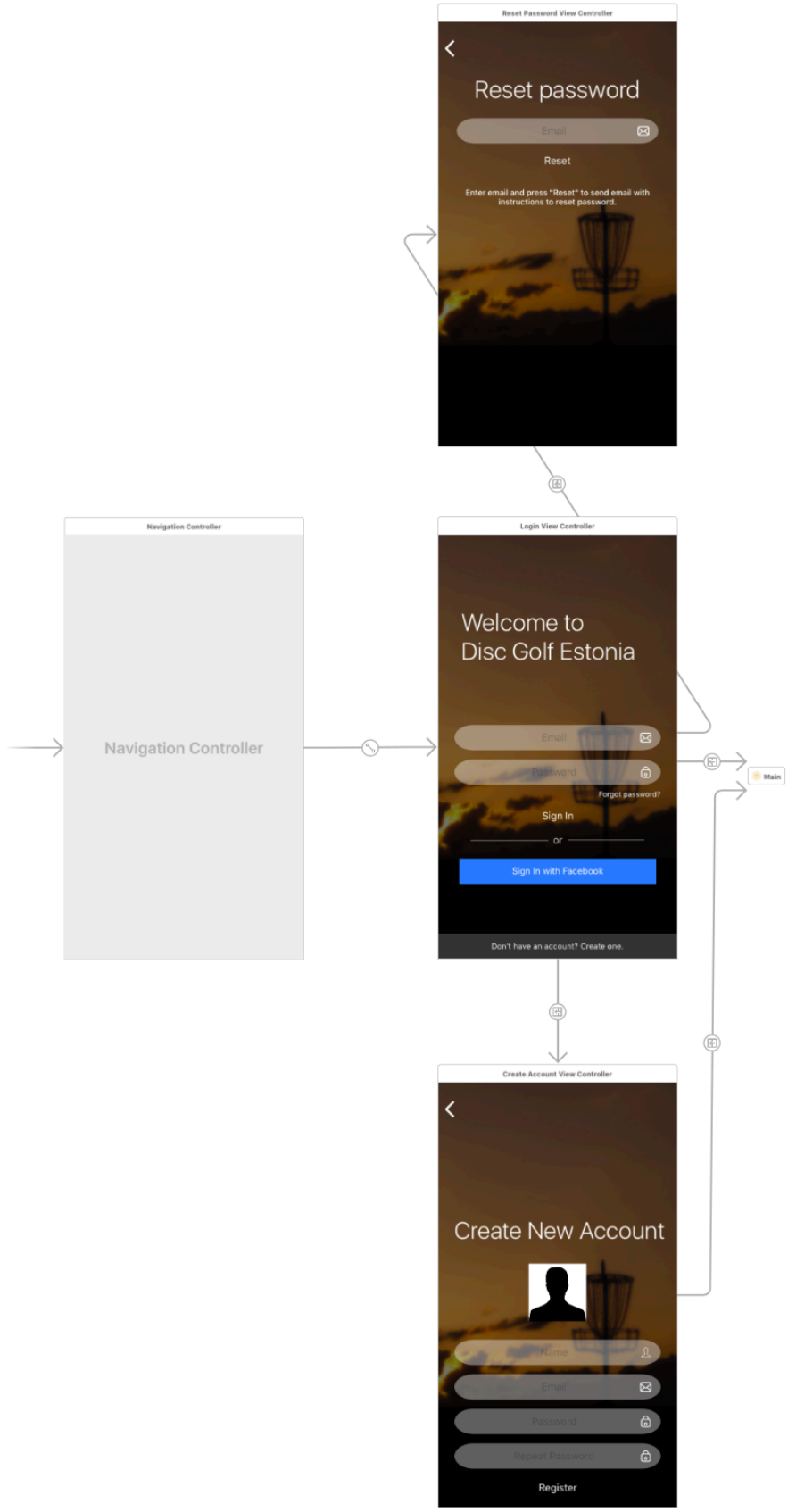


Figure 15. Storyboard: Login.

Courses storyboard:

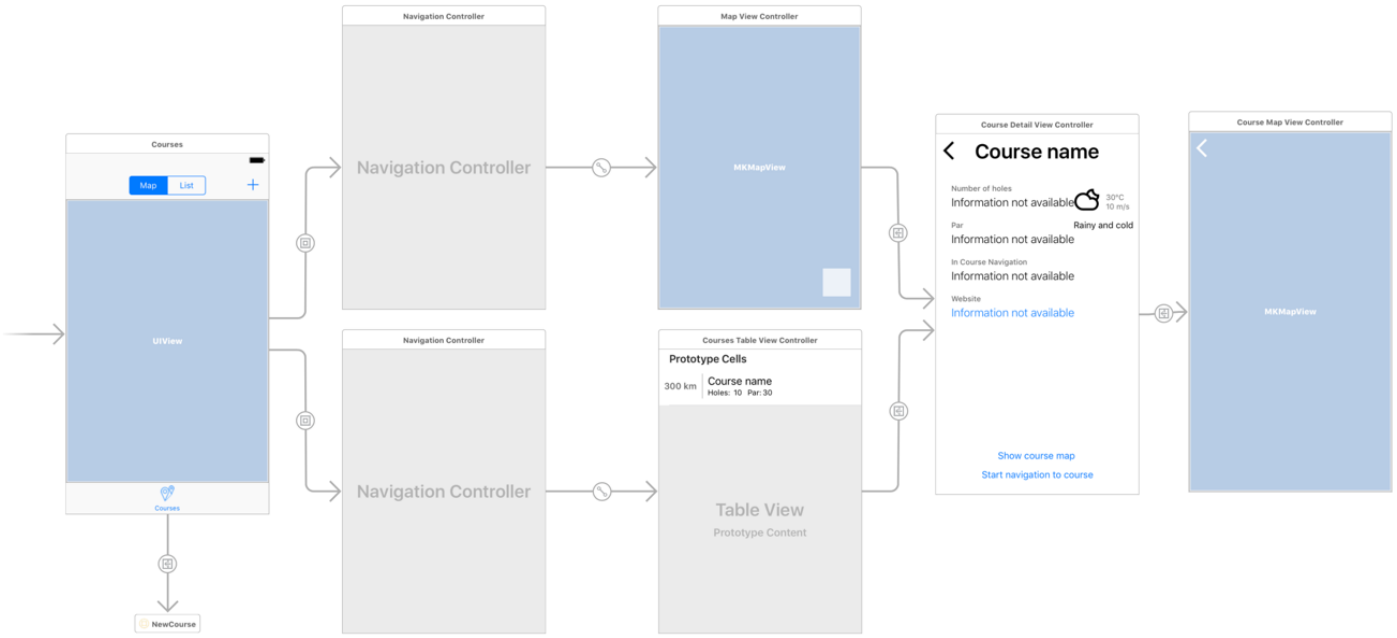


Figure 16. Storyboard: Courses.

New course storyboard:

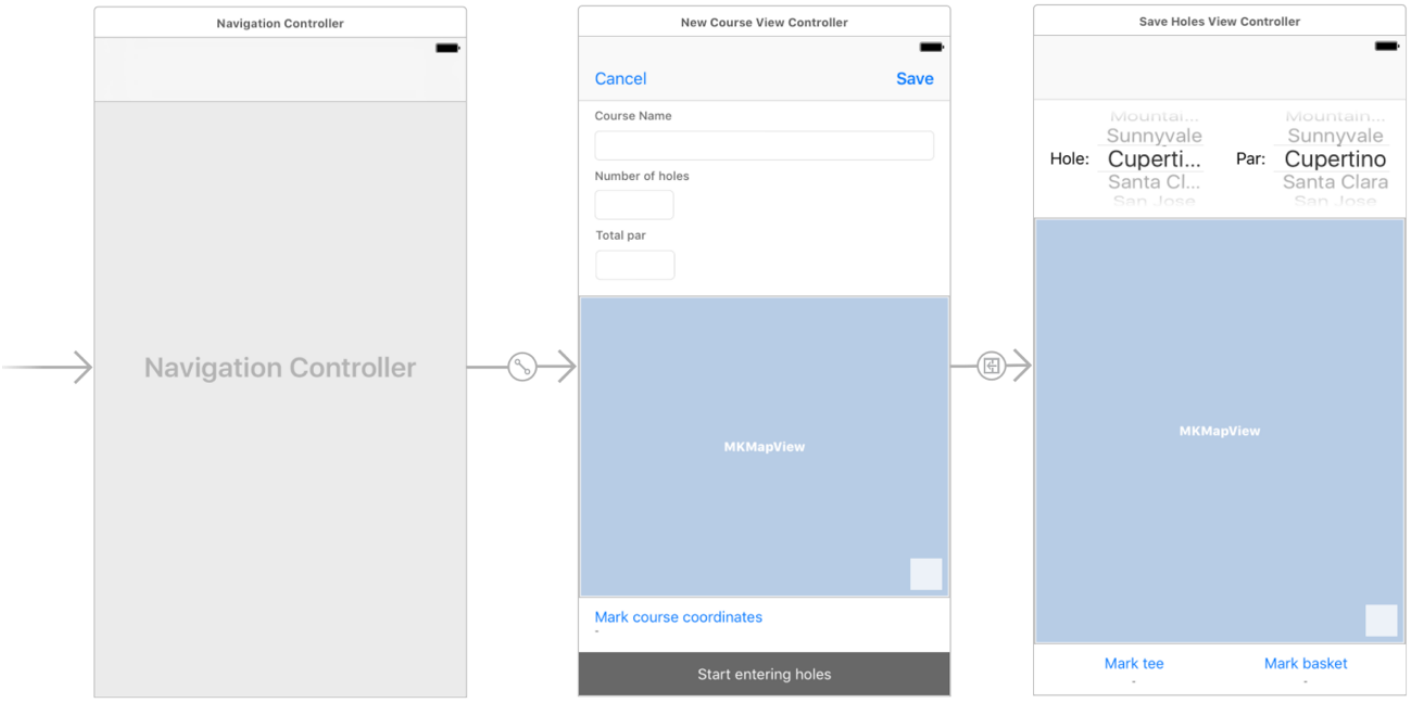


Figure 17. Storyboard: New course.

Game storyboard:

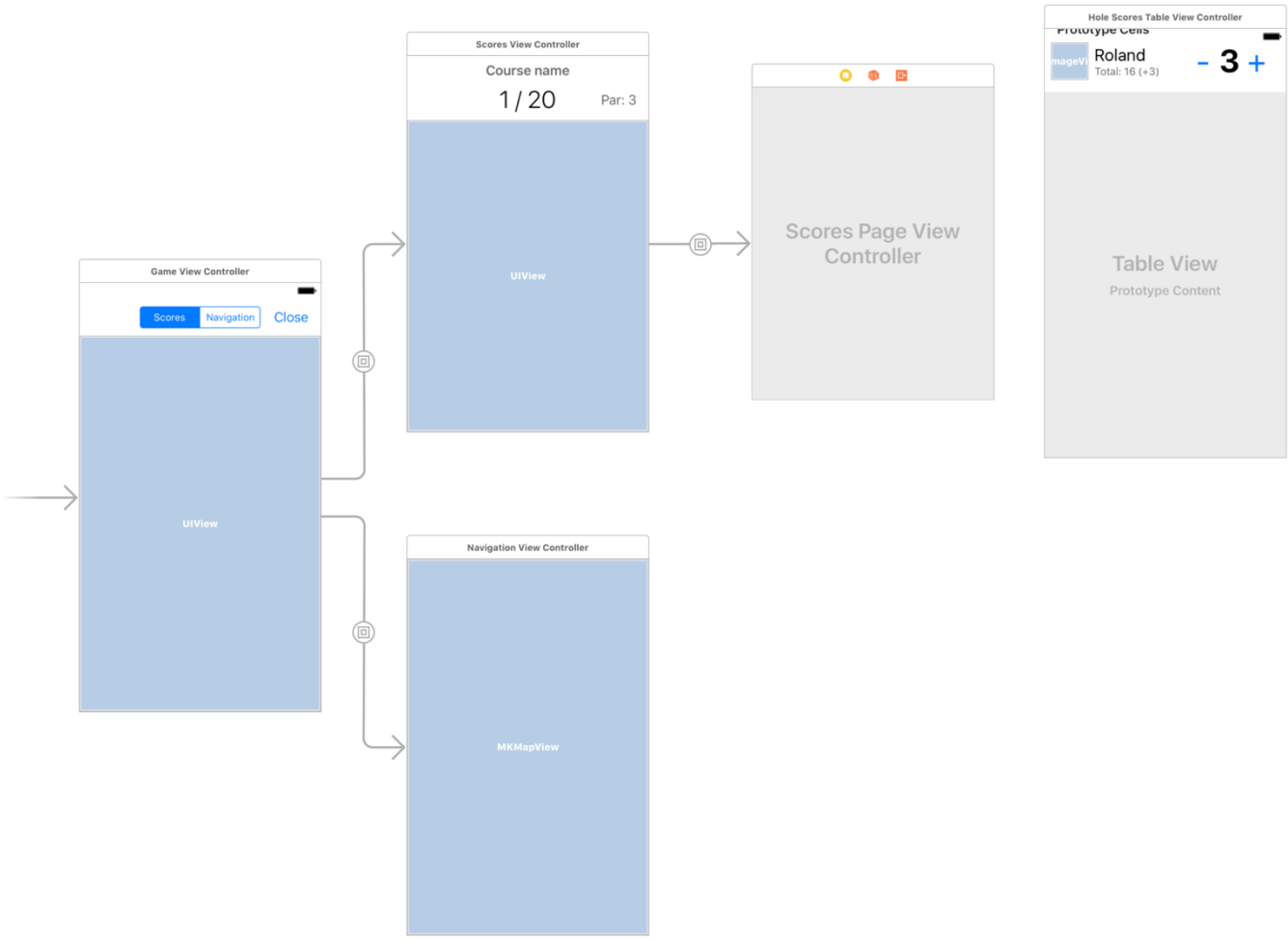


Figure 18. Storyboard: Game.

Play storyboard:

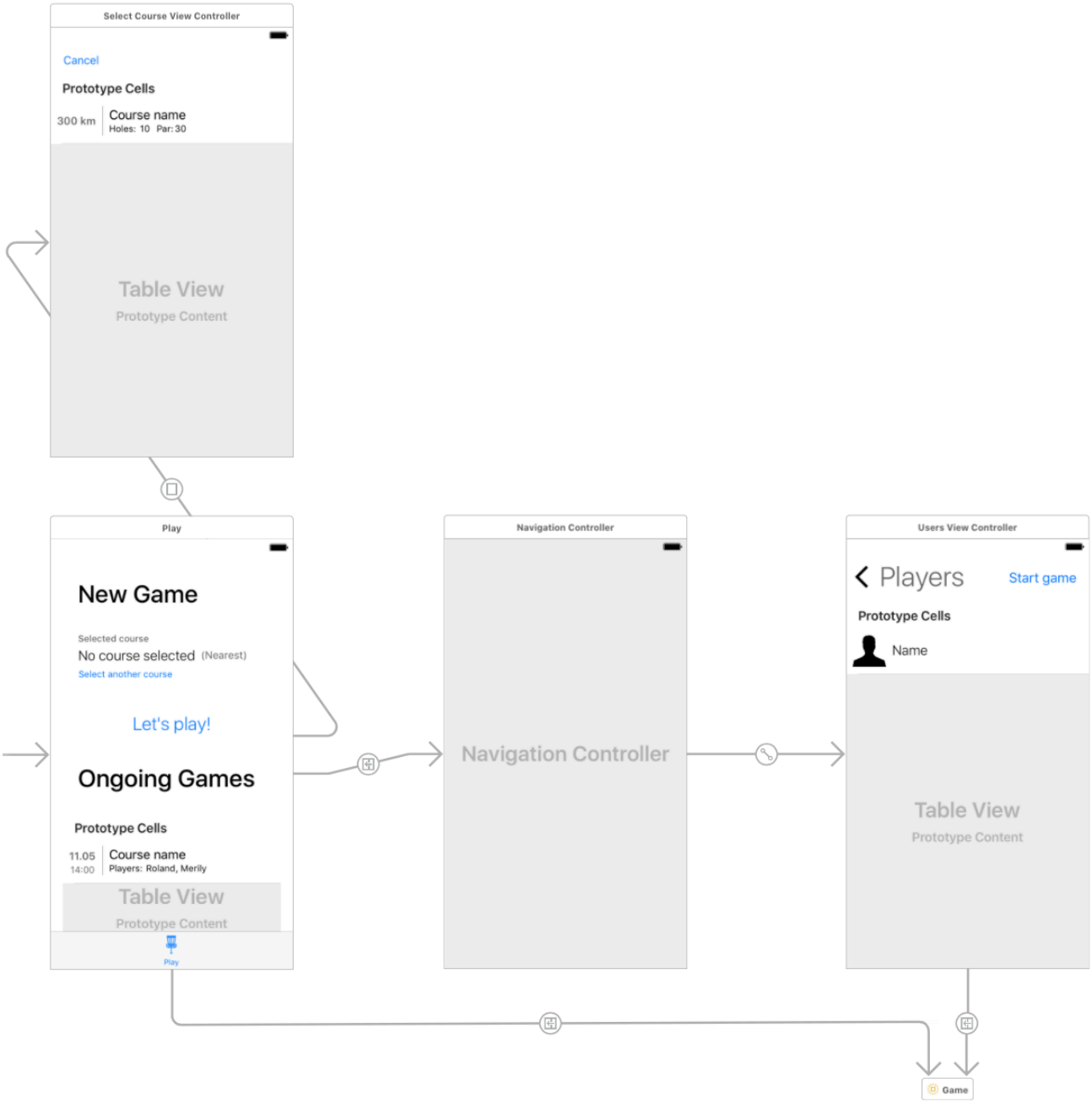


Figure 19. Storyboard: Play.

Profile storyboard:

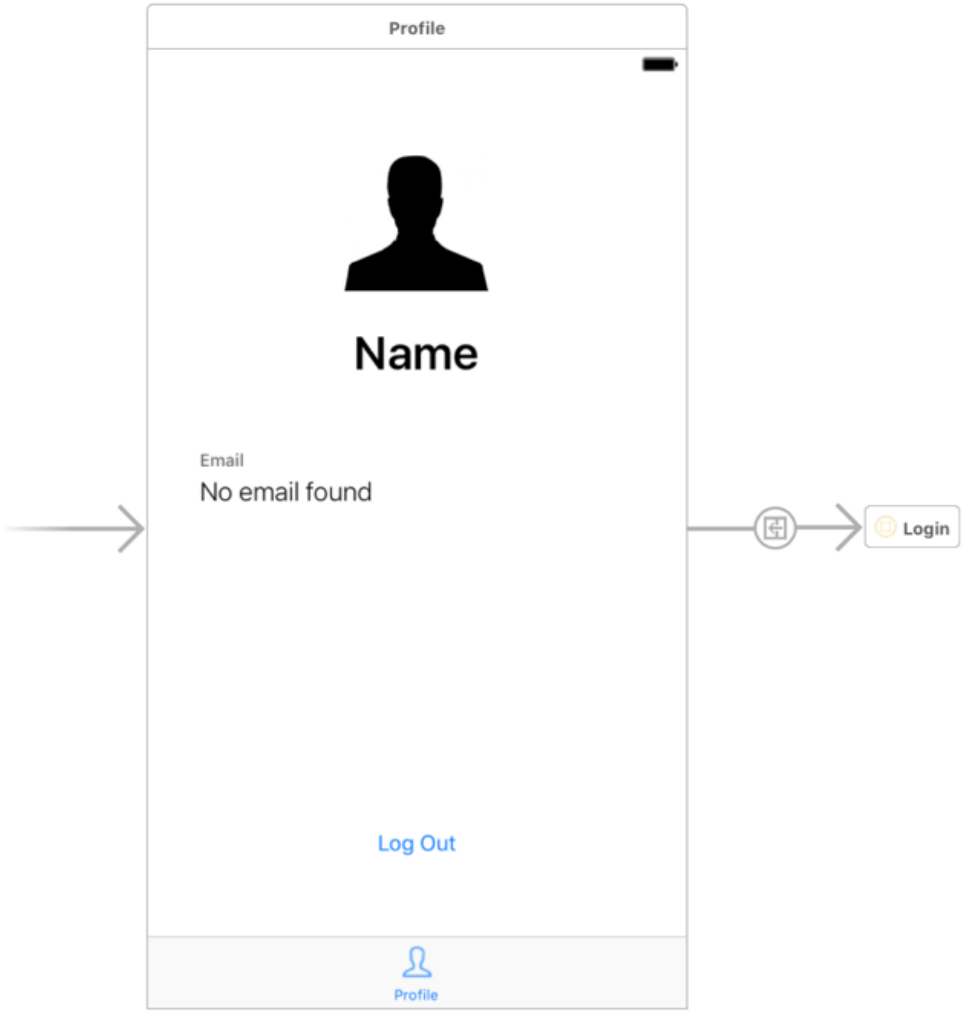


Figure 20. Storyboard: Profile.

# Appendix 2 – Activity diagrams of main processes

Main processes in the app described with conceptual activity diagrams:

1. Creating new account

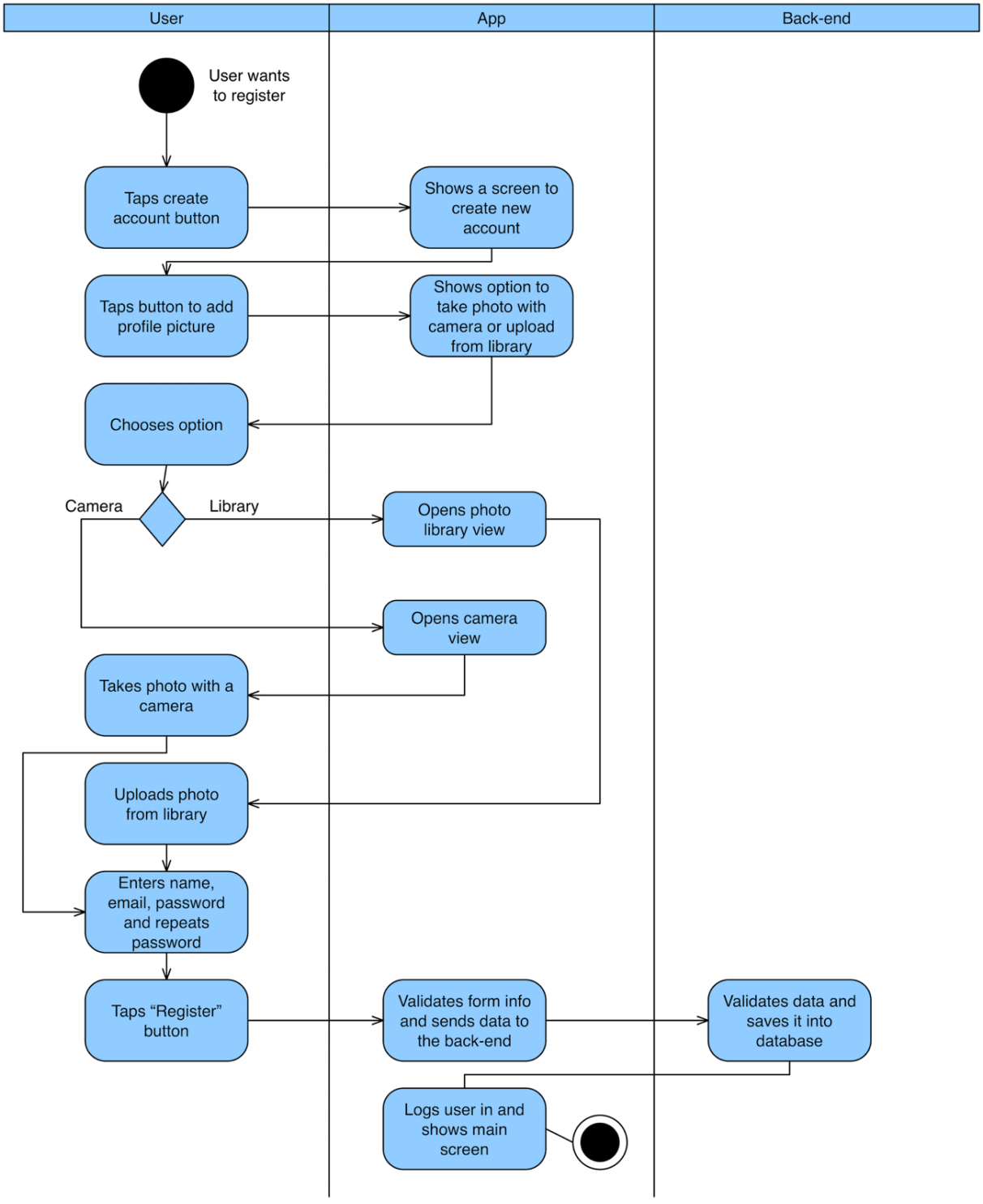


Figure 21. Activity diagram: Creating new account.

2. Signing in

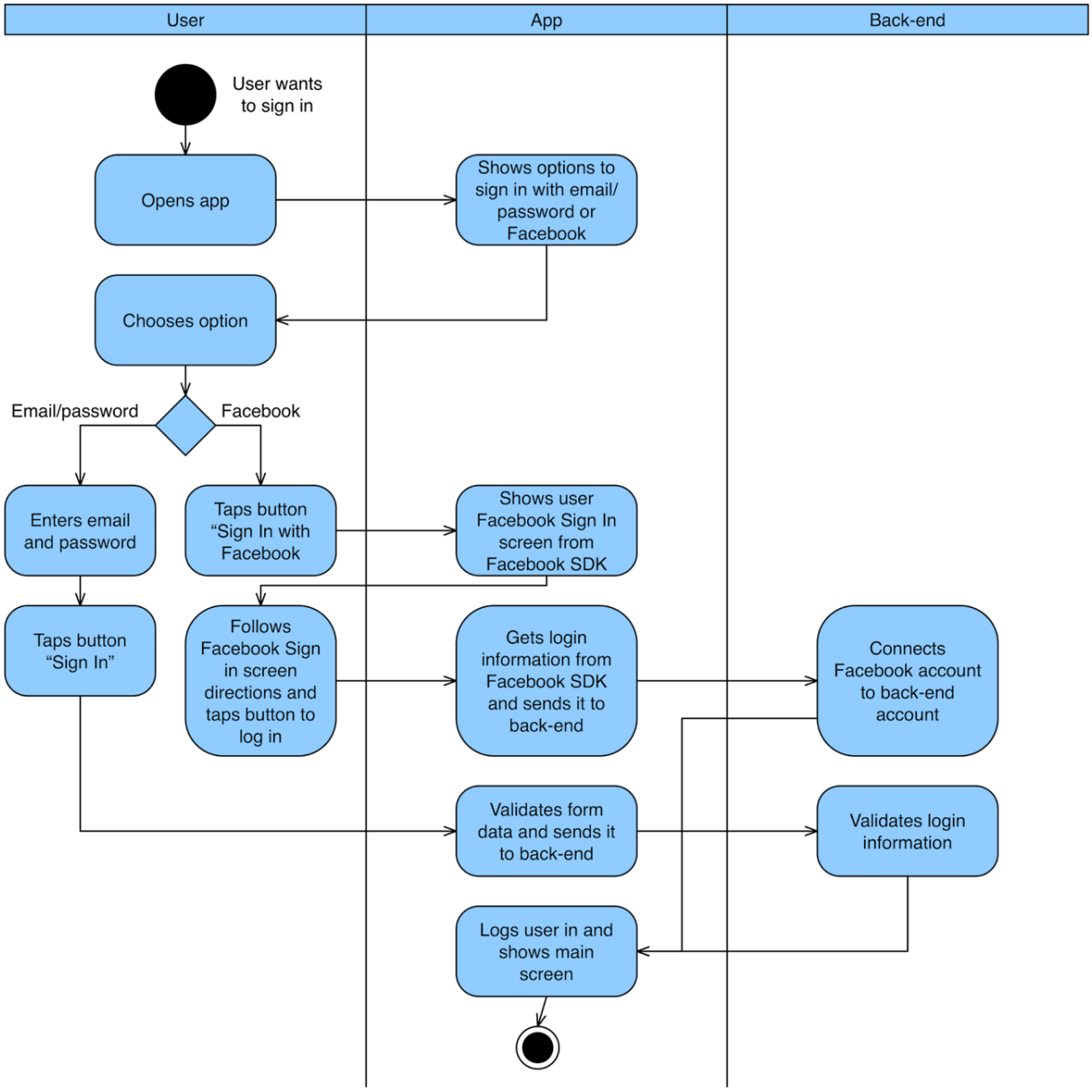


Figure 22. Activity diagram: Signing in.

3. Adding new disc golf course

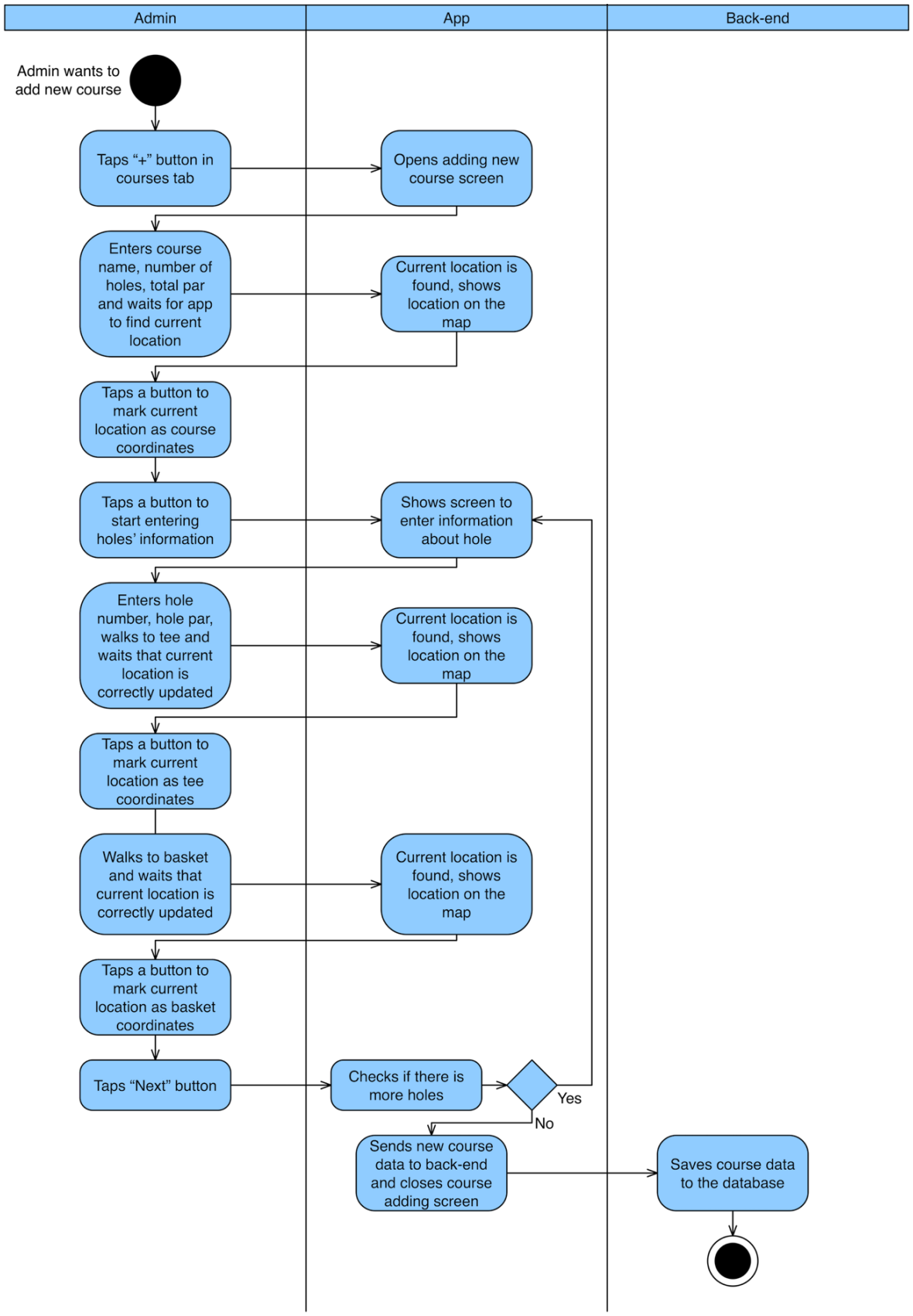


Figure 23. Activity diagram: Adding new disc golf course.

4. Getting to the disc golf course

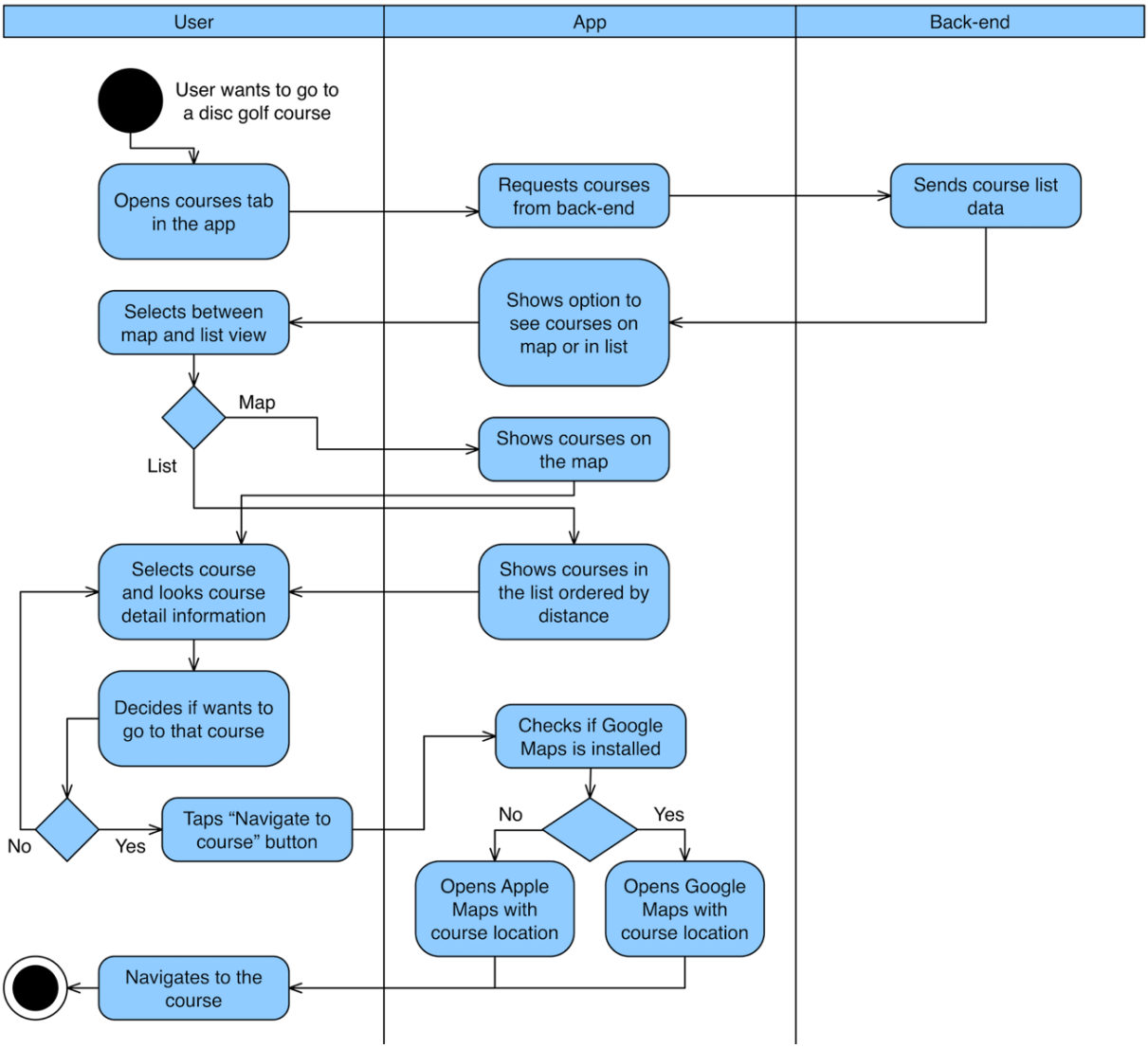


Figure 24. Activity diagram: Getting to the disc golf course.

5. Starting a new game

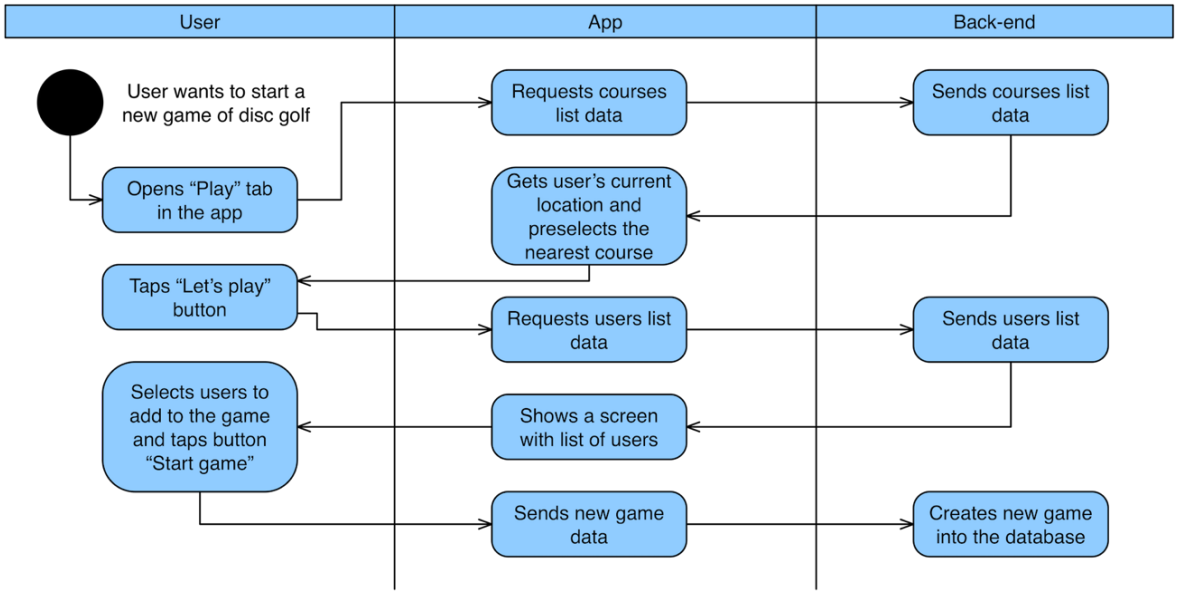


Figure 25. Activity diagram: Starting new game.

6. Playing the game

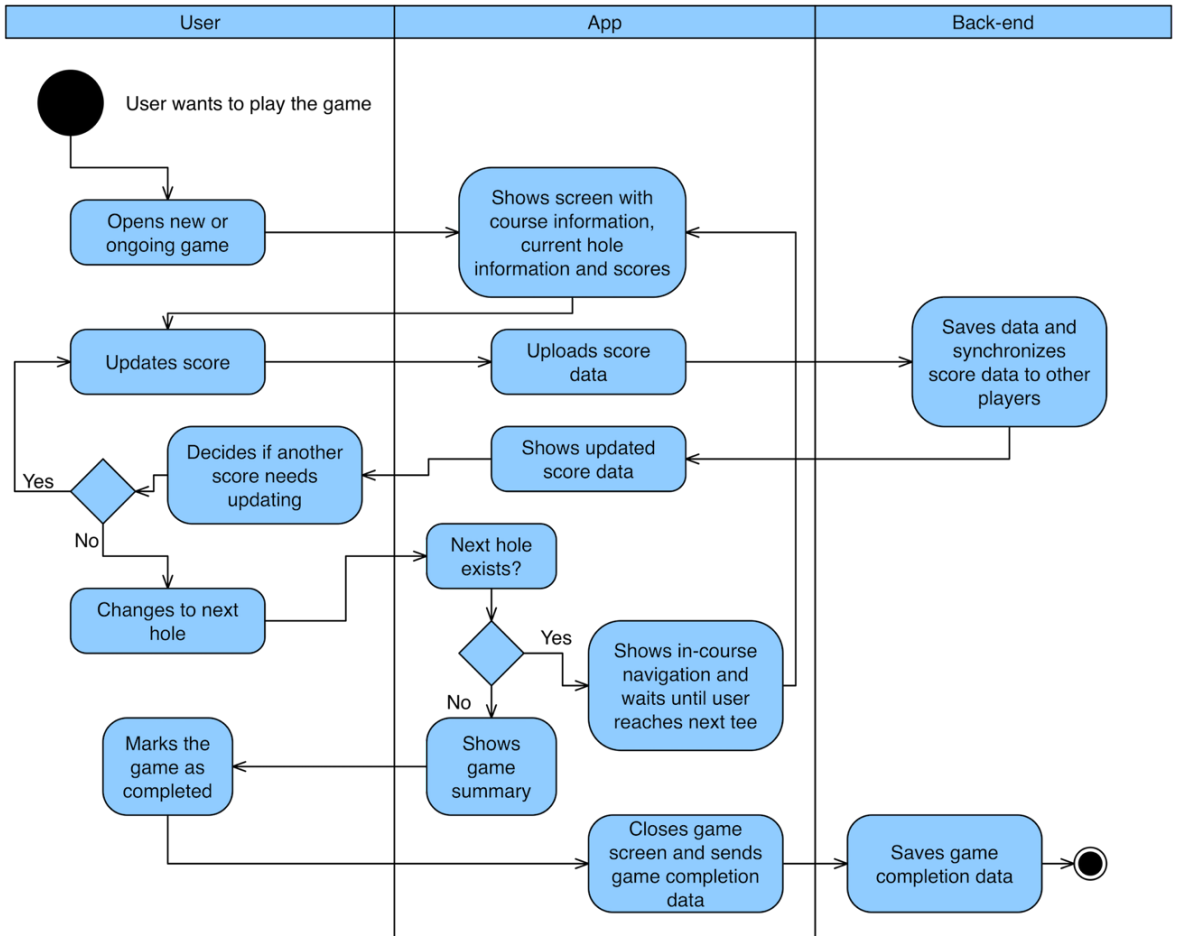


Figure 26. Activity diagram: Playing the game.

# Appendix 3 – Source code repository

The source code of the iOS disc golf application is publicly available at https://github.com/RolandTalvar/disc-golf-estonia.