# 1. Project Plan

## 1.1 Aim

The aim of the project is to build an iOS Application and Web Server that allows users to create and join sport events, with a build in chat function that allow user to message each other, and a discover function that lets you find people near you that you can invite to events.

## 1.2 Objectives

* The client application will be developed for iPhone
* The client application will be able to communicate with a Web Service
* The client app will not communicate with remote database directly
* The web server will store data on PostgreSQL
* The client application will be developed using Swift
* The web server will be developed using Node.js with sails.js framework
* The steps of developing system will be described in Project Report
* The process of the system development will be described in Process Report.

# 2. Time Estimates

The product was developed for the client MoveSport. They wanted the app ready in AppStore in spring (April 2015), so initially we sat down and planned how we could achieve that. It was decided that the app will be released over two releases, in that way the client would have a presentable app ready by 1st of April. The first one would be a simple version of the app that still could work, and the second would contain more advanced functions. So together with the client we decided the following three deadlines

1. **March 1st**

The complete flow & design of the app. The initial home screen implemented where a user can sign-up/login, both manually or with a facebook account using Facebook SDK.

1. **April 1st**

The first release of the app. Function such as

* Login/Sign up
* Show events on a map, and information when clicking on them
* Create/join/delete events
* Show “My Events” (events created and joined by the user)
* My Profile page where user can see favorite sports
* Edit profile and update favorite sports functionality

1. **May 1st**

* Chat function, ability for users to chat with participants for the same event
* Filter function to filter events on the map by category
* Discover function
  + Discover people within a radius selected by the user, then the possibility to invite them to events created/joined.
* Push notification for: Chat & when inviting user to event

1. **June 1st**

Project & Process Report. Bug fixes - if any.

# 3. Group Members

The group consists of Alexandru Vasile, Ibrahim Yildirim and Juraj Petrik.

# 4. SWOT Analysis

## 4.1 Juraj Petrik

|  |  |
| --- | --- |
| Strengths: Quick at learning new programming languages and frameworks.  Experienced in Javascript and Node.js | Weaknesses: Bad at estimating how much time things take  Postponing more difficult tasks |
| Opportunities: Make a great project with clean code  Learn about Sails.js framework | Threats: Missing deadlines.  Not spending enough time on the project. |

## 4.2 Alex Vasile

|  |  |
| --- | --- |
| **Strengths:**  Strong work ethic  iOS development  Drive to learn  Hard working | Weaknesses: Time estimation  Does not handle stress well. |
| Opportunities: Learn Swift  Improve design skills  Learn about useful OS libraries | Threats: Spending too much time to make things perfect  Too heavy workload from external  sources |

## 4.2 Ibrahim Yildirim

|  |  |
| --- | --- |
| **Strengths:**  Thinking out the box  Good with UI  Getting things done fast  Quick Learner  Great problem solving skills | Weaknesses: Using long time on UI instead of functionality  Postponing  Time Management |
| Opportunities: Building project from scratch  Will learn the new frameworks, languages and technologies quicky | Threats: DiggitApp & Other projects  Will postpone tasks and not complete tasks on deadline  Too much CC |

# 5. Group Policy

The group members worked together in their internship and various other projects, therefore they understand each other’s work habits really well. All of them strive to create the best possible product and to finish it on time. Here we outline some of the main policies the members agreed upon:

* Meet at the agreed time
* Commit code often and regularly for every new feature
* Do not commit code that breaks the build
* Document your work
* Discuss technical choices with your peers
* Value the input of other group members

# 6. Implementation Phase

## 6.1 Version Control

To start working on the implementation, it is essential to set up version control system. Version control is used to store and distribute files and source code for project development.

Since we had a lot of experience with it, we decided to use Git [1].

Our remote private repositories are hosted on github.com. We have divided our project into two separate repositories:

1. Swift application

git@github.com:alexcosmin/SportLook.git

2. Node.js Api Server

git@github.com:alexcosmin/SportLookBackend.git

## 6.2 Time Management

For keeping track of the time and the work needed to be done within a given time in order to meet the client deadlines, the team decided to use an agile method. A small team of disciplined and highly skilled developers is likely to succeed regardless of which agile method they use, so with that in mind we used some artifacts of SCRUM. The main elements we used from SCRUM were the backlog,stories and sprints. We, however, decided to omit the burn-down chart to measure progress, in order to not spend too much time on the process itself but on the planning and implementation of the application. As for SCRUM roles the product owner is Ibrahim Yildirim, our SCRUM master is Juraj Petrik. We simply used enough elements from SCRUM, so that we knew which tasks needed to be done when, and if some were not completed on time, we would work nights and weekends to make sure they were before the deadlines.

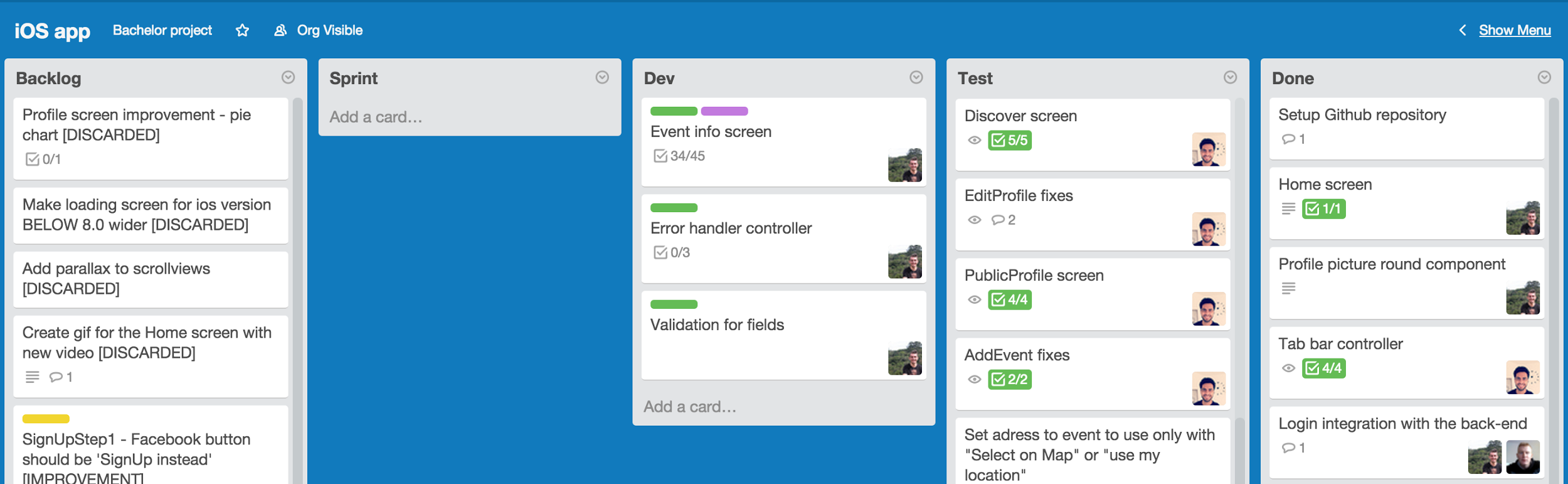
We decided to go make the first deadline (March 1st) in one 3-week sprint and then the next divide each release in two 2-week sprints, making a total of 5 sprints. By dividing the workload that needed to done in two sprints, the team would have a overview of how much there would be left, when half of the time was passed, and if it was on the right track for the release.

Before each sprint we follow these steps

* If it’s not the first sprint we spend one to two hours to do backlog grooming, hold a short sprint retrospective and write a summary of the retrospective in the process report.
* We take product backlog items and split them into smaller tasks, and add them to the sprint backlog.
* We take tasks from sprint backlog and decide which group member will be responsible for the task execution.

During the sprint

* We add the tasks to our board in trello.com
* According to the task’s status, the group member who is responsible for the task, moves them across five columns: Backlog, Sprint, Dev, Test, Done



Below you can see the entire product backlog for our project

Priority 1 = Deadline March 1st

Priority 2 = Deadline April 1st

Priority 3 = Deadline May 1st

|  |  |  |  |
| --- | --- | --- | --- |
| Nr | Title | Description | Priority |
| 1 | Design application flow & wireframes | Designing the main flow of the app | 1 |
| 2 | Design system architecture | Design domain system architecture | 1 |
| 3 | Setup iOS development environment | Setup Xcode IDE and create initial project with version control | 1 |
| 4 | set up web servers database | Create database for web server with object relation mapping, populate tables with test data | 1 |
| 5 | Design Graphical User Interface | Design all the screens for the application | 1 |
| 6 | Implement welcome screens | Implement Home screens in iOS | 1 |
| 7 | Implement API Functionality to support home screens | Create models and controllers to support user CRUD functionality | 1 |
| 8 | Implement Authentication | Create controllers and Sails.js policies to properly implement authentication process | 1 |
| 9 | Design and Create iOS Models | Design and create the models in Xcode | 2 |
| 10 | Design and Create API Services | Design the initial data flow and communication with the web server | 2 |
| 11 | My Profile Tab | Implement the screen for User Profile, together with edit profile and update favorite sports. | 2 |
| 12 | My Events Tab | Implement the screen for MyEvents, Add event & Event info. | 2 |
| 14 | Search Events Tab | Implement MapKit integeration and calls to the webserver to get events | 2 |
| 15 | Add serialization/deserialization of JSON Objects | Implement for API models JSON serialization and deserialization and test if correct values are retrieved and sent from web server. | 2 |
| 16 | Discover Tab | Implement Discover screen, Public profile. | 3 |
| 17 | Invite user to event | Implement functionality to get current events and invite a user to it | 3 |
| 18 | Event Chat | Chat functionality for users to chat with each other if they are participating in the same event | 3 |
| 19 | Push Notification | Create and implement Apple Push notification service in the project. | 3 |
| 20 | Filter | Filtering function for in the “Seach Events” Tab. | 3 |
| 21 | Release to App Store | Create App record in Member Center & iTunes Connect | 1 |
| 22 | Integrate Client App with Parse | Integrate Client app with Parse | 3 |
| 23 | Integrate Web Servier with Parse | Integrate Web Server with Parse | 3 |

Initial Meeting with client

First meeting with the client happened on February 9th. After the meeting we discussed and concluded that it would be best to release the app over two times in the App Store. There could be a production ready version ready by April 1st and a upgrade with more functions by May 1st. The client agreed and the team started making the product backlogs.

Sprint 1 (February 9th - March 1st)

In this sprint there was focus on the UX design of the application. We would have to have some design mockups ready by the end, so the android developers would build a application that was similar to the iOS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Backlog Title | Description | Resp | Est |
| 1 | Design application flow & wireframes | Designing the main flow of the app | I & A | 30 |
| 2 | Design system architecture | Design domain system architecture | I, A, J | 10 |
| 3 | Setup iOS development environment | Setup Xcode IDE and create initial project with version control | A | 10 |
| 4 | Set up and test web servers database | Create database for web server with object relation mapping, populate tables with test data | J | 20 |
| 5 | Design Graphical User Interface | Design all the screens for the application | I & A | 80 |
| 6 | Implement welcome screens with facebook integration | Implement Home screens in iOS | I & A | 10 |
| 7 | Implement API Functionality to support home screens | Create models and controllers to support user CRUD functionality | A & J | 20 |
| 8 | Implement Authentication | Create controllers and Sails.js policies to properly implement authentication process | J | 60 |

Sprint Retrospective

A lot of time during this sprint was spent on designing the layout. The client received the finished mockups before a couple days before the sprint ended and had some comments.

He wanted us to use a different color palette but the overall design was approved.

For the Web Service, setting up the authentication properly took quite a lot of time. The project was set up on GitHub, and the team also got introduced to the programming language Swift, it was interesting and we were excited to work more with it.

Sprint 2 (March 1st - March 15th)

In this sprint we are going to start the initial screen of the application after the user has logged in such as: My Profile, Add Events, My Events. Also we will finish the login functionality so it’s integrated with the web server.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Backlog Title | Description | Resp | Est |
| 9 | Design and Create iOS Models | Design and create the models in Xcode | A | 30 |
| 10 | Design and Create API Services | Design the initial data flow and communication with the web server | J | 90 |
| 11 | My Profile Tab | Implement the screen for User Profile, together with edit profile | I | 70 |
| 12 | My Events Tab | Implement the screen for My Events, Add event | A & I | 90 |

***Sprint Retrospective***

A lot of time during this sprint was spent to dig deeper into Swift and Sails.js, and setting up the models of the application. It was a good start for the app, and we managed to make a big progress for this first sprint. We are on the right track, and looks like the app will be ready for the 1st release soon.

On the server side some problems were encountered because of the Waterline database adapter. The waterline adapter is still in constant development process and not all features are well developed and tested. For example our domain data models are working with a lot of relations between models and in if we need to perform CRUD operations for the parent model which contains children in several levels, such a feature in Waterline was not yet implemented. To overcome those issues Janis had to find some workaround and later on he submitted a new feature request for Waterline. If Waterline will implement this feature in further releases, then we will have to update our server side code. Overall the sprint went well and the tasks were successfully implemented, tested and documented.

**Sprint 3 (March 16th - March 31st)**

In this sprint we will have to have a releasable version of the app ready. Also we will have to make the app record in Member Center and iTunes Connect. We will be working on the MapKit framework in order to add map functionality in the application, and the a clustering component that will make it look good when there is a lot of locations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Backlog Title | Description | Resp | Est |
| 11.3 | My Profile Tab | Function to update Fav Sports | I | 20 |
| 10.4 | Design and Create API Services | Integrate AWS for image upload | J | 90 |
| 12.2 | Event Info | Implement the screen for showing event info | A | 35 |
| 14 | Search Events Tab | Implement MapKit integration and calls to the webserver to get events | I | 40 |
| 15 | Add serialization/deserialization of JSON Objects | Implement for API models JSON serialization and deserialization and test if correct values are retrieved and sent from web server. | A | 40 |
| 21 | Release to App Store | Create App record in Member Center & iTunes Connect | I | 20 |

***Sprint Retrospective***

During the 3rd sprint we focused on having a App Store ready application. A lot of the time went with testing the UI and making sure it was stable, without crashing.

On the client application we encountered some problems with UIScrollView because of auto-layout, and making it fittable for each of the different screen sizes.

On the server side some minor problems were encountered because of the Waterline database adapter. Overall the sprint went well and the tasks were successfully implemented, tested and documented.

The Sprint ended with a meeting with the client, to demonstrate the app to the client. First release sent to review for App Store!

**Sprint 4 (April 1st - April April 14th)**

In this sprint we need to start on working on the upgrade for the next deadline which is in one month. During this sprint we will make proof of concept with some open source libraries, and try to have a look at Parse.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Backlog Title | Description | Resp | Est |
| 16 | Discover Tab | Implement Discover screen, Public profile. | I | 20 |
| 17 | Invite user to event | Implement functionality to get current events and invite a user to it | I | 30 |
| 19 | Push Notification | Create and implement Apple Push notification service in the project. | A & J | 40 |
| 20 | Filter | Filtering function for in the “Search Events” Tab. | I | 15 |
| 22 | Integrate Client App with Parse | Integrate Client app with Parse | A | 60 |
| 23 | Integrate Web Server with Parse | Integrate Web Server with Parse | J | 20 |

***Sprint Retrospective***

During the sprint we managed to get a connection between our system and Parse. We created a push notification development certificate in Apple’s member center, and set it up, so we are now able to send notifications to the app from the backend. It is not fully implemented and we still need to hand what happens when the user presses on the different notification, in order for the right screen to show up.

From the back-end point of view there was a lot of issues with Amazon Web Services which we used for the image upload. We kept encountering different errors, so there was a lot of time spent to fix those issues.

**Sprint 5 (April 15th - 1st May)**

In this last sprint we will have to get the app done completely. As it looks in the beginning of the sprint it looks very feasible, and we have managed to make a really scalable app, that we can keep adding functionality to, without the biggest issues.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | Backlog Title | Description | Resp | Est |
| 18 | Event Chat | Chat functionality for users to chat with each other if they are participating in the same event | A & J | 80 |
| 19.2 | Push Notification | Making Push notifications work with the chat | A & J | 40 |
| 21.1 | Update the app in App Store | Create the required certificates for the update and update the record in iTunes Connect | I | 5 |
| 22.1 | Integrate Client App with Parse | Setting the users to the corresponding channels in order for the chat to work | A | 20 |

***Sprint Retrospective***

During the sprint we finally achieved the full flow of the app as the client initially wanted. The data flows flawlessly between the Web API and the client, and the features are very well tested.

Some problems we encountered were coordinating the chat and push notification functionality with the Android development team. It was crucial for the iOs and Android application to send out and handle chat messages in the same way so the chat works regardless of the devices used. The Android developer had a lot of problems understanding the quite simple flow of the functionality but with a lot of help and support from us he overcame the difficulties in the end.

We have developed our project to satisfy the objectives of the project and client requirements.

# 7. Personal Reflections

The overall outcome of the project has been very satisfactory. We spent a lot of time and effort to make it just right.

One big problem that surfaced was the need to support and communicate with the team that was developing an Android version of our application. The client MoveSport hired a danish company to develop the the Android version. This danish company decided to outsource this development to a single developer in India. Now when this Indian developer encountered a problem, usually with our WebService, he would contact the danish company, who would contact the client, who would in turn contact us. Obviously this was very unproductive communication workflow that made our job the more frustrating. Unfortunately, we didn’t have any influence over this decision of the client.

# 8. Literature

[1] Git Visual Guide https://docs.google.com/document/d/1HjMsr9hNIZLyuwQzS25oskyN30\_sb5j2VOogCr7tI7M/edit?usp=sharing