Lappeenranta University of Technology

Code Camp (Autumn 2012) "Agile Java Development by Capgemini"

-

Final Report

Alexander Fürthmaier (g0404515)
Esa Hiiva (g0398207)
Mehar Ullah (f0392591)

1 Table of Content

2 Table of Figures & Tables

3 Introduction

This is the final report to complete the Code Camp on Agile Java Development by Capgemini (Fall 2012). We are a team of three members and we developed a workload management system from the very beginning using the GRAILS Framework. This framework implements the MVC pattern and was a very good approach to complete the task within a short time.

4 Method and Tools

4.1 MVC Pattern with GRAILS

As mentioned in the introduction we used the MVC Software pattern with the GRAILS framework to implement our application. The MVC pattern implies the use of Models, Views and their Controllers to implement the complete application.
The Model classes represent the necessary data with its attributes that can be stored in a database. A very nice feature is the pre-implemented and very easy to use database based on the Hibernate Framework. So there is no need to set up a database of our own. With commands provided by GRAILS we were able to store and retrieve data from the database.
The View classes are responsible for the look and feel of the application. Basically these files are plain HTML combined with CSS and some JavaScript. GRAILS provides some additional <g: > tags for the view classes to easily interact with the controller.
Controllers are hosting the complete logic and methods. Content is dynamically created and rendered into the view classes.
As soon as you are used to these special GRAILS tags and syntax and the whole MVC concept, it is easy to design a web-application with its logic encapsulated in the controllers, the data represented by the models and at least presented to the user via the view classes.

4.2 Agile Development with the SCRUM method

Scrum is an iterative and incremental agile software development method for managing software projects and product or application development. The main characteristics of the SCRUM method used in our development process were the daily SCRUM meeting and the several sprints with its backlog. Within this code camp, we tried to implement three SCRUM sprints. (For every week one sprint with its goals). According to that, at the beginning of each week we developed the sprint backlog. The daily SCRUM was used to discuss the progress and to talk about difficulties to get an overview of the state of the actual sprint. In the next two pages we present our sprint backlogs. These backlogs are based on the user stories mentioned in the next chapter.

Table 1: Spring 1 backlog

#	Task	state	responsibility	Comments
1	GUI - Login Screen	done	Mehar	
2	GUI - Home Screen	done	Alex	
3	GUI - Create Forecasts	partly done	Esa	Missing dynamic table

4	GUI - Browse Forecasts	done	Esa	
5	GUI - Manage Activity Types	done	Esa	
6	GUI - Reports	done	Alex	
7	Data structure - Domain classes	partly done	Esa	skipped monthly based estimation
8	Views and controllers	done	Mehar	
9	Login / Logout Logic	done	Mehar	
10	Logic - Workload Estimation	done	Alex	
11	Reports - Data extraction	partly done	Esa	ARVE, URVE done, not COR
12	Reports - Calculations	partly done	Mehar	ARVE, URVE done, not COR
13	Reports - Presentation	done	Alex	

Table 2: Sprint 2 backlog

#	Task	state	responsibility	Comments
1	improve navigation and overall GUI	done	Alex	
2	add availability information to reports	done	Alex	
3	implement roles	partly done	Esa	
4	Different views to the system depending on the role	done	Esa	

5	implement team based report	skipped to sprint 3	-	
6	implement special queries to database	skipped to sprint 3	-	
7	implement over 100% of workload alarm to forecasts	done	Mehar	
8	Implement possibility to get reports for weekly and yearly intervals.	done	Mehar	

Table 3: Sprint 3 backlog

#	Task	state	responsibility	Comments
1	Team based report	open	-	to do
2	implement special queries to database	open	-	to do

4.3 Version Control

Since we were a team of three members working on the same time on different tasks, we decided to use a SVN-Repository to synchronize our code. We used a code.google.com repository and the subclipse plugin for eclipse. At the beginning of each day, after the daily scrum we started with the synchronization of our data (update and commit) in order that all of us can be able to work on the actual version of our system. Merging our code regularly using SVN helped us to fix some bug right away instead of fixing plenty of bugs at the end due to merging huge line of code (LOC).

4.4 Information sharing

At the beginning of the code camp we started to create documents in the traditional way using office software on our computers and sending these documents via e-mail to other group members afterwards. We quickly recognized that we need another solution to improve our collaboration. We decided to use Google Drive to improve the efficiency of our team-work. To sum up we mainly used the Google Drive to achieve the following goals:

- Easily sharing of information and files within our group
- Parallel working on documents (e.g. backlog, user stories or even this document)

5 Overview of the System

In this chapter we give an overview of the system. First of all we are going to present our user stories developed at the beginning of the code camp. These user stories are based on the requirements document. In the next step we describe the structure of the system and its main functionalities.

5.1 User Stories

Based on the requirements document we formulated several user stories in order to be able to interpret the requirements of the system from the user's point of view. Since we had in mind that the time will be short to complete all the tasks, so we decide to complete it in three sprints, we divided the user stories in 3 categories (of course based on the importance mentioned in the requirements document, mandatory or beneficially). Each category should be - more or less - completed within one sprint.

Priority 1: Core functionality

- As a user I will be able to see and edit forecast on browser
- As a user I will be able to create, (edit and remove) an activity types
- As a user I will be able to create, edit and remove an activity for distinct consultant
- As a user I will be able to create weekly or monthly workload recordings for different timeframes
- As a user I will be able to update the data at same time as other user is updating data
- As a user I want my current forecast be loaded when I open forecast view
- As a user I want be able to create a report

Priority 2: Resource management

- As a user I want to be able to manage users and teams
- As a user I want to be able to link users to a teams

Priority 3: Special features

- As a user I will be able to go over 100% workload but I will be notified if I do so.
- As a user I will be able to inspect forecasts which take holydays into account.
- As a user I want be able to track all changes

This was our first guess how our sprints could look like. In the end we changed our sprint goals in some points but this is exactly the advantage of the agile software development. You are able to change things during the implementation.

The user stories above leaded us to define most of the functionalities we want to implement.

5.2 Structure of the System

5.2.1 Homescreen

After the login process the user gets to his home screen where all possible tasks are offered via big buttons. We identified the following three major tasks the system should provide to its users:

- Forecasts functionality
- Report functionality (based on the forecasts made by consultants)
- Management section

These major tasks are provided by the homescreen which is displayed to the user right after the login. We call it the "Forecast section", "Report section" and "Management Section". Depending on the role of the user who is logged in to the system the user can either view all of the sections (e.g. as manager) or just part of them. By clicking on one of these sections you get access to its associated functionality.

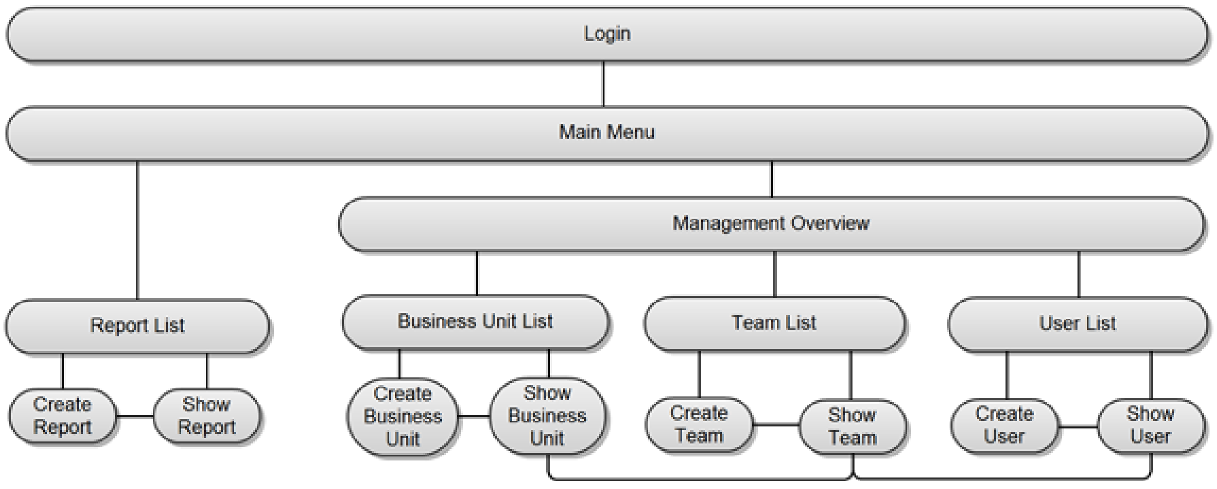


Figure 1: Screens accessible for user in Director role

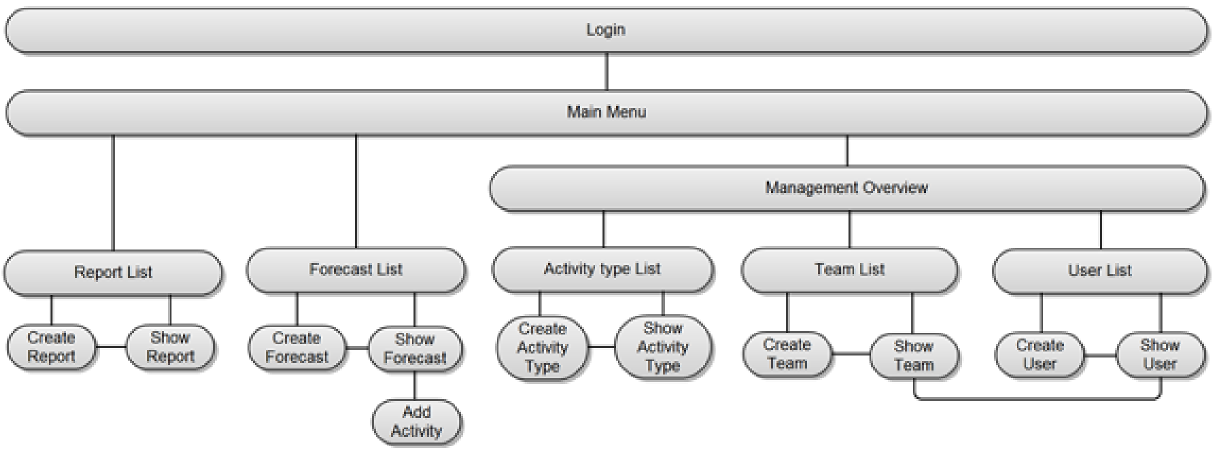


Figure 2: Screens accessible for user in Manager role

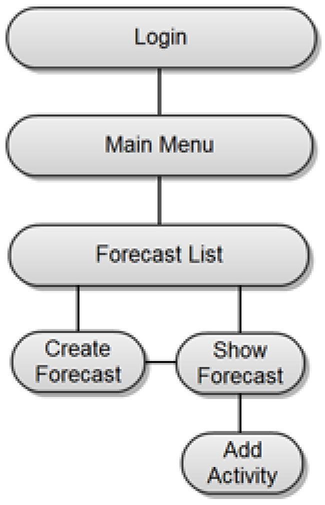


Figure 3: Screens accessible for user as a Consultant role

5.2.2 Navigation-bar on top

With the horizontal navigation bar right below the banner the user is offered additional functionality of the system. At any level of the system the user is able to do the following actions:

- Go to previous page
- Go to Homescreen
- Logout from the system

Further the navigation bar on the top is offering additional functionality to the user depending on the section the user is right in. For example, in the forecast section the user is offered functions like “create forecast”

5.2.3 Navigation-bar on bottom

In addition to the navigation bar on top, distinct functionality is offered to the user via the navigation bar on the bottom. In this bar operation to change items are offered. You will mainly find the following two function on the bottom of the system:

- Edit an element
- Remove an element

5.3 Description of main functionalities

5.3.1 Authentication

Authentication is implemented using traditional username and password combination. User must input valid username and password to gain access into system. Login process is same for users using different roles, but only users with Director or Manager role can create new users. All screens of the application can use the authentication information which is saved to the session. Currently authentication is only checked in the system level at the login process. Authentication checks are not yet implemented in every screen of the system.

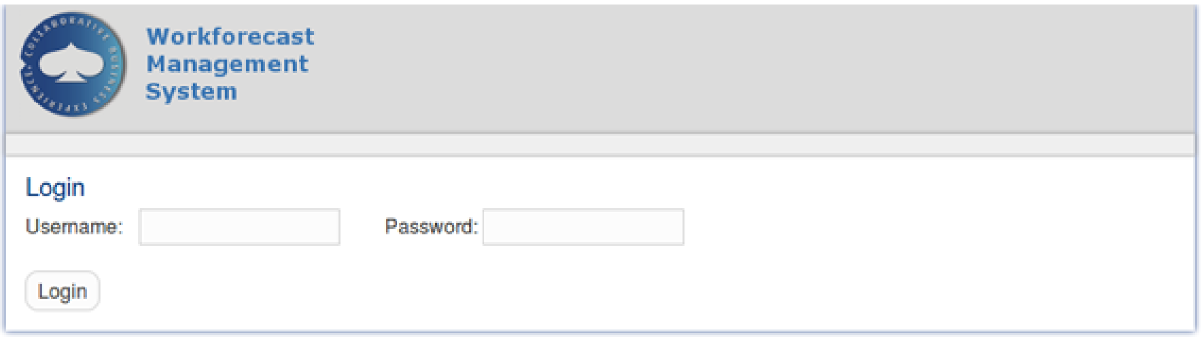
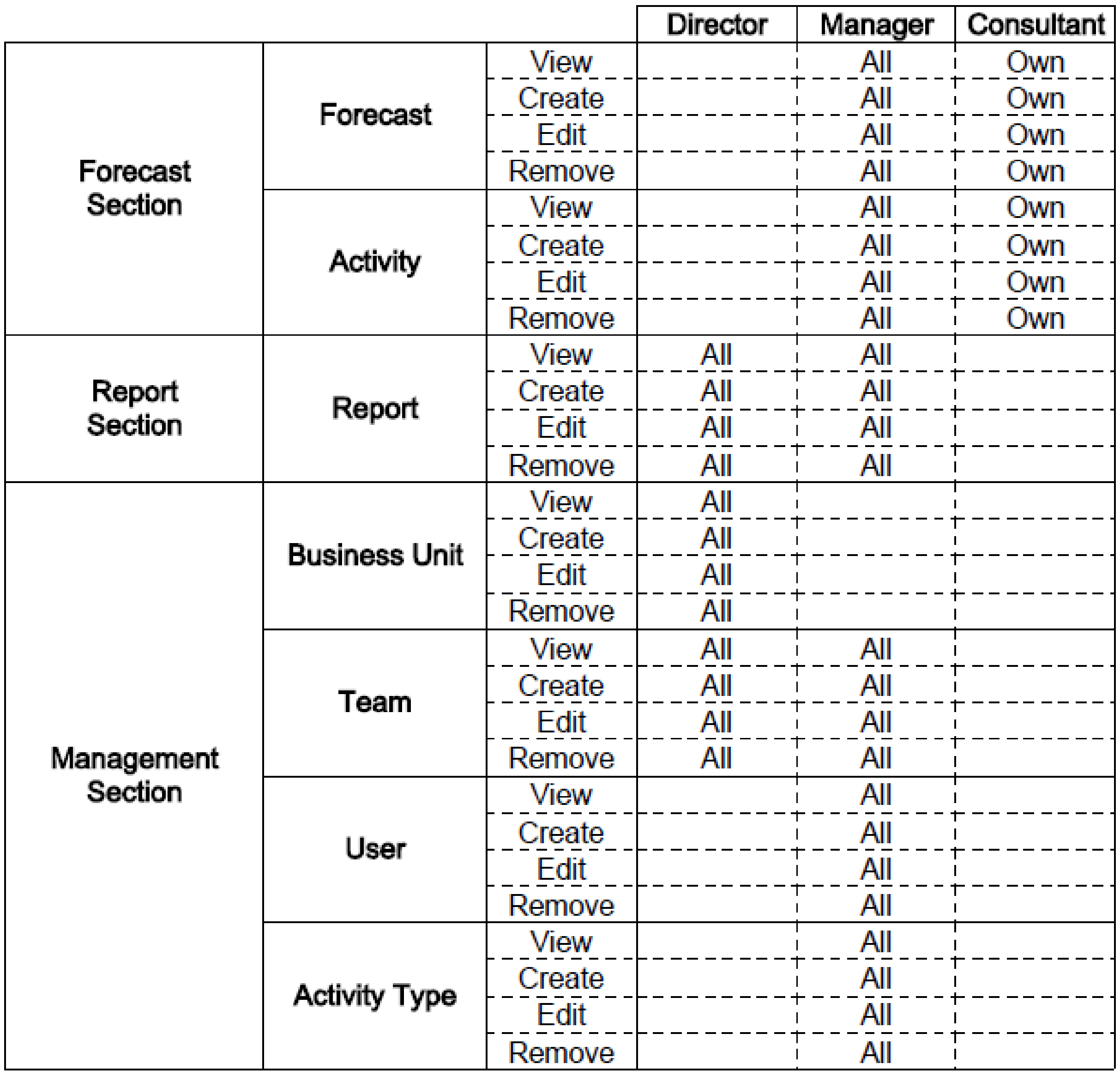


Figure 4: Login screen

5.3.2 Authorization

User role is used for authorization in the home screen and in the management overview screen. Authentication is also used in forecast list screen. When user is logged in as a Consultant role, he/she can see only his/hers own forecasts. User with a Director or Manager role has always full access to data from all the other users. Manager role can be scoped down in future updates, so that manager is linked to team(s) and has authorization to see only data from his/hers own team(s).



			Director	Manager	Consultant
Forecast Section	Forecast	View		All	Own
		Create		All	Own
		Edit		All	Own
		Remove		All	Own
	Activity	View		All	Own
		Create		All	Own
		Edit		All	Own
		Remove		All	Own
Report Section	Report	View	All	All	
		Create	All	All	
		Edit	All	All	
		Remove	All	All	
Management Section	Business Unit	View	All		
		Create	All		
		Edit	All		
		Remove	All		
	Team	View	All	All	
		Create	All	All	
		Edit	All	All	
		Remove	All	All	
	User	View		All	
		Create		All	
		Edit		All	
		Remove		All	
	Activity Type	View		All	
		Create		All	
		Edit		All	
		Remove		All	

Figure 5: Permissions of different roles

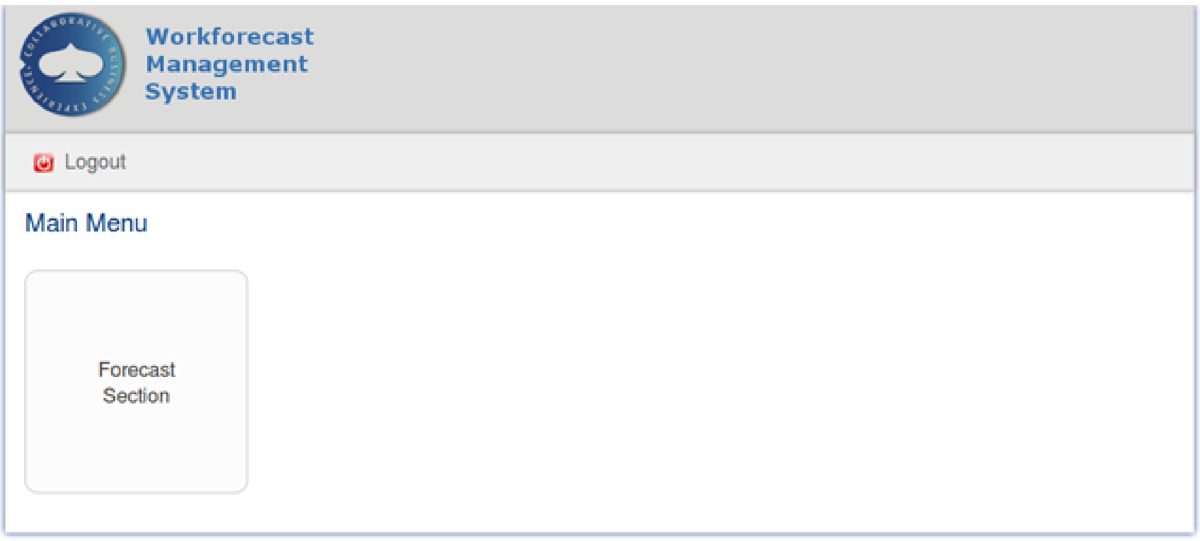


Figure 6: Homescreen for Consultant role

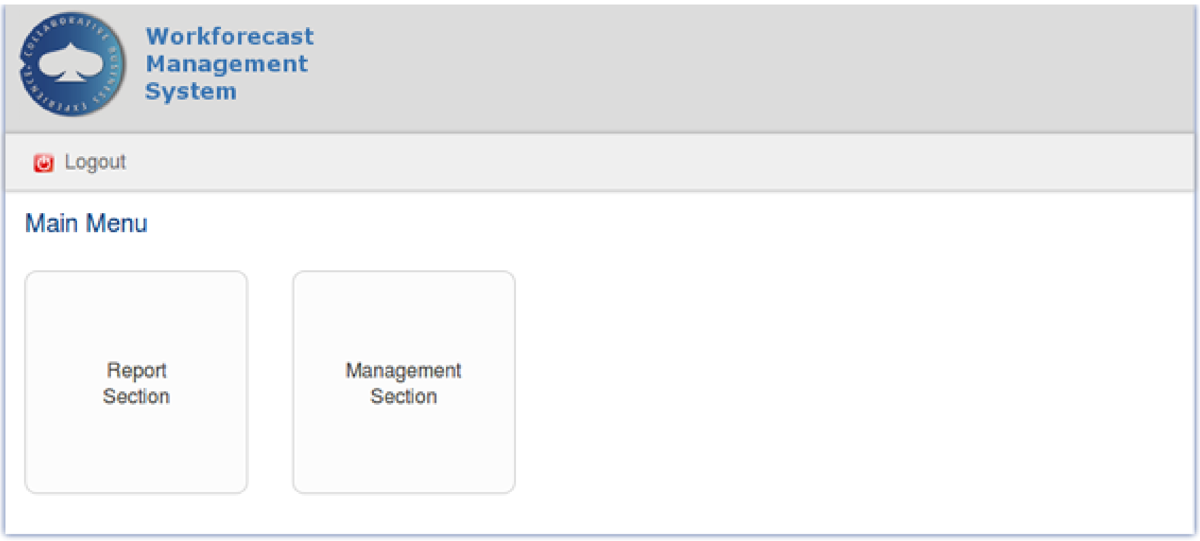


Figure 7: Homescreen for Director role

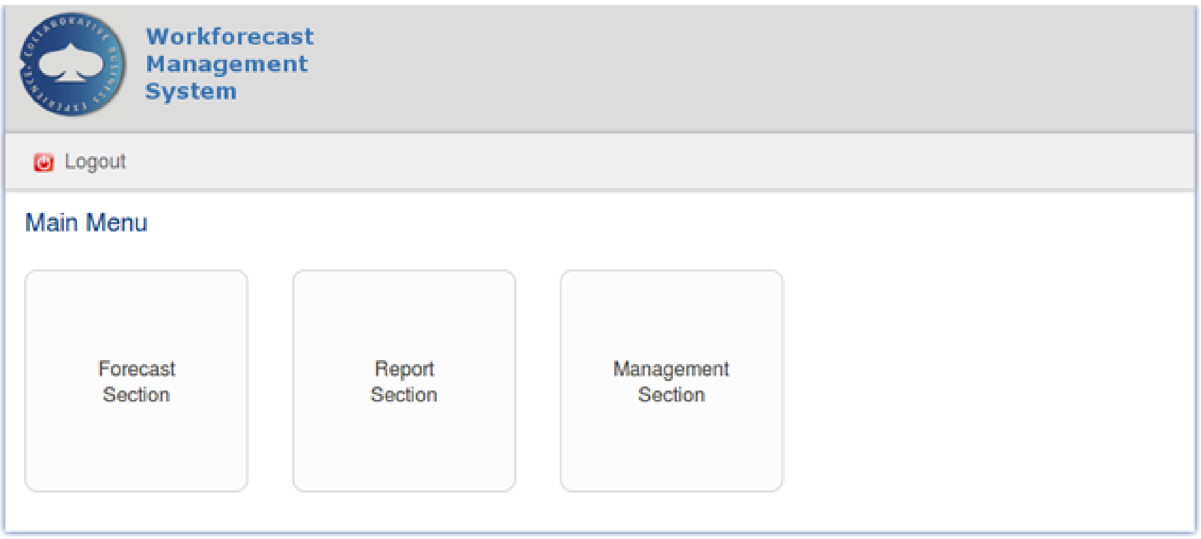


Figure 8: Homescreen for Manager role

5.3.3 Management Section

Management section contains activity type management and staff management which are divided into three parts. Staff is managed using business units, teams and users. Every user belongs to a team and every team belongs to a business unit.

Management section has different options for Director and Manager roles. Only a Director can create business units and assign teams to business units. Managers operate at the next level and can create teams and team members.

Manager can create new activity types and edit existing activity types. Activity types are dynamic and selected options are affecting calculations in the reports. Manager has to create activity types before consultants are able to create new forecasts.

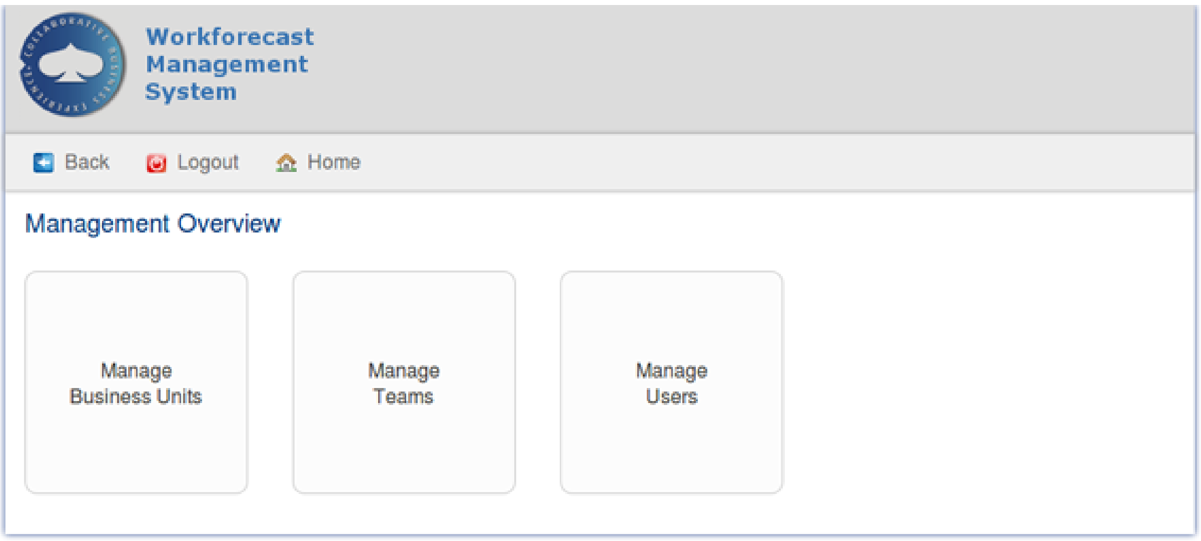


Figure 9: Management overview for Director role

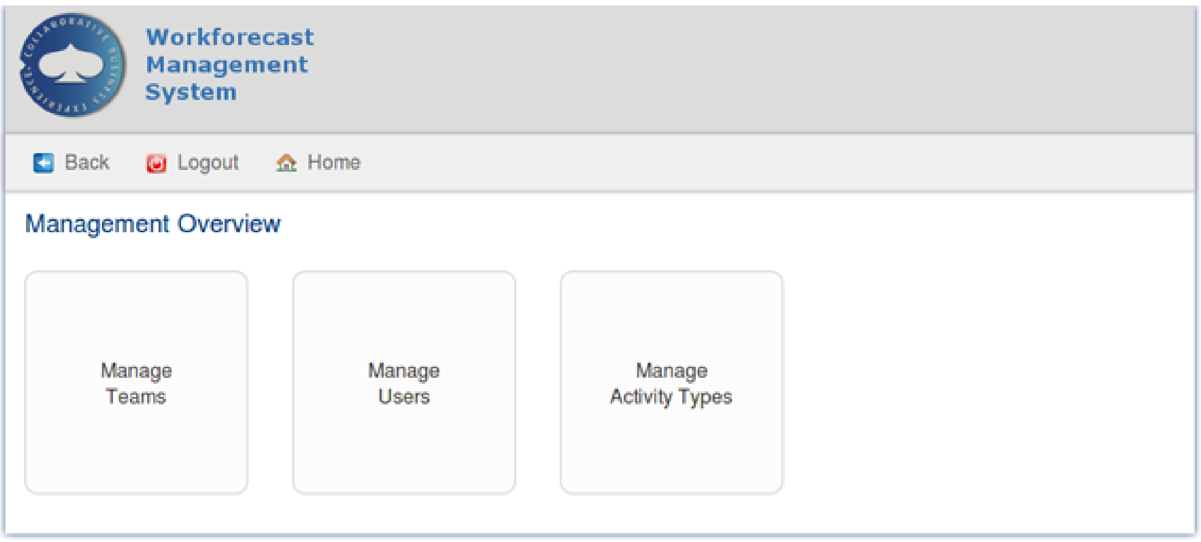
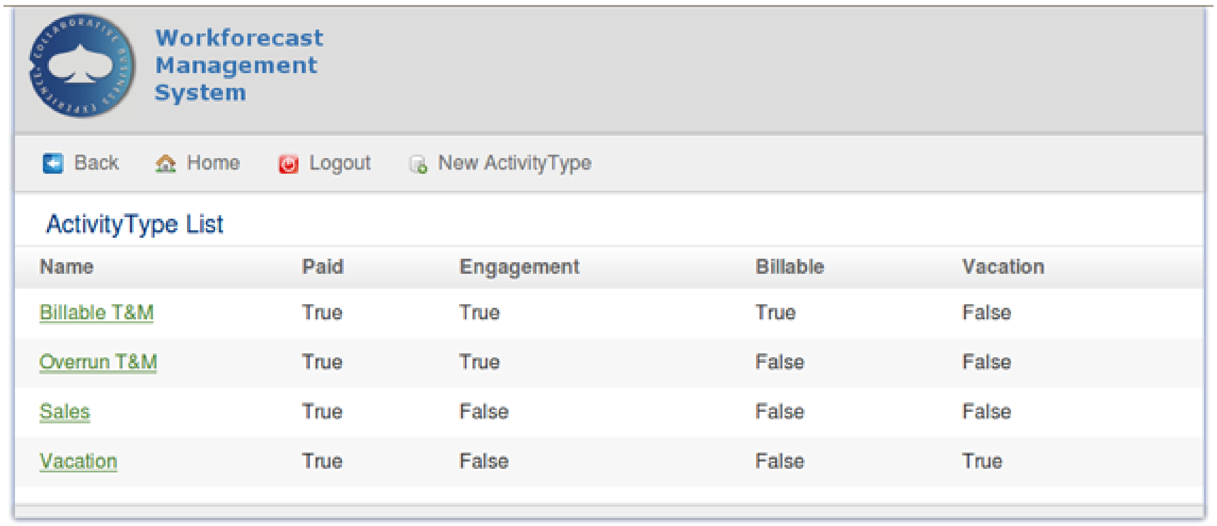


Figure 10: Management overview for Manager role

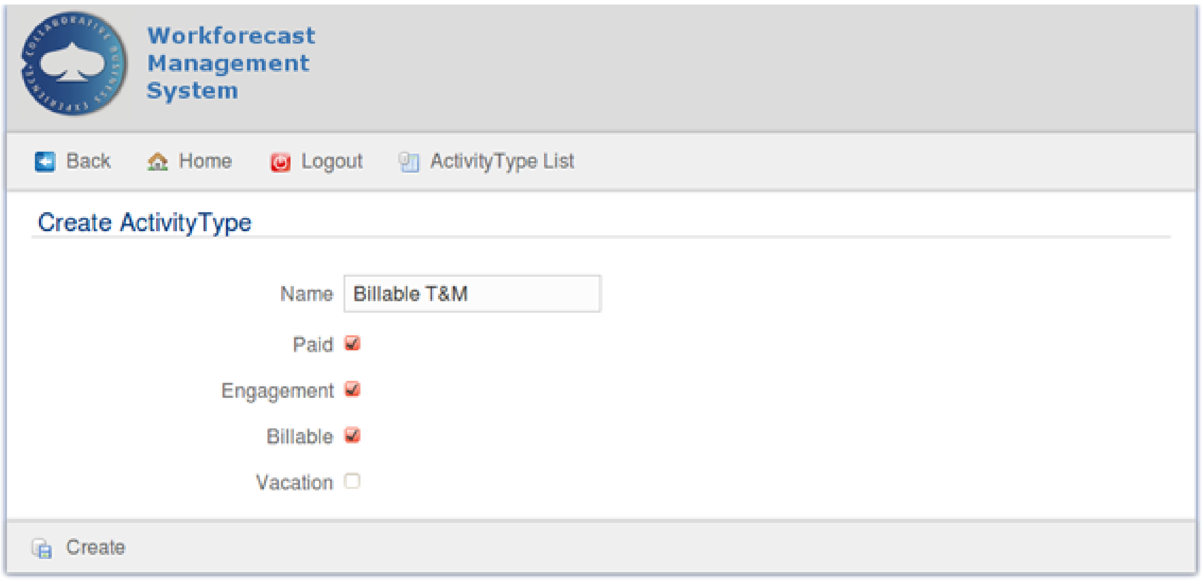
All four manageable items under the management section have same set of screens. There is always a list screen, create screen, edit screen and show screen. Create and edit screens use same _form.gsp fragment for rendering form fields and there is not much difference in those screens. Scaffolding was used to create views and fragment based create and edit screens were created automatically in that process. Fragments make development and maintenance easier, because form fields have to be changed only in one place. Next there is a sample from different screens.



ActivityType List

Name	Paid	Engagement	Billable	Vacation
Billable T&M	True	True	True	False
Overrun T&M	True	True	False	False
Sales	True	False	False	False
Vacation	True	False	False	True

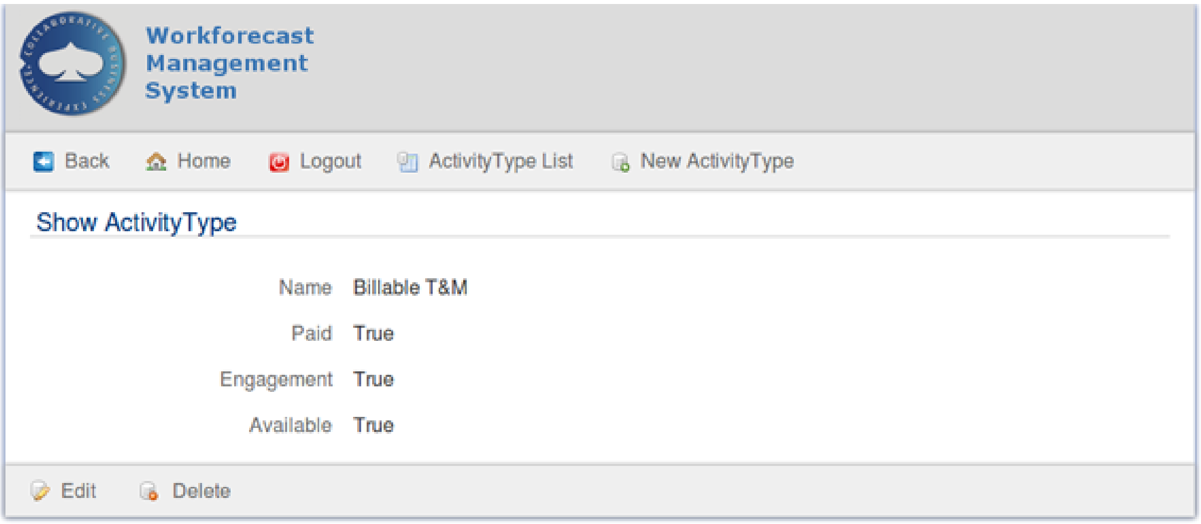
Figure 11: Activity Type list screen



Create ActivityType

Name	Billable T&M
Paid	✓
Engagement	✓
Billable	✓
Vacation	☐

Figure 12: Activity Type create screen



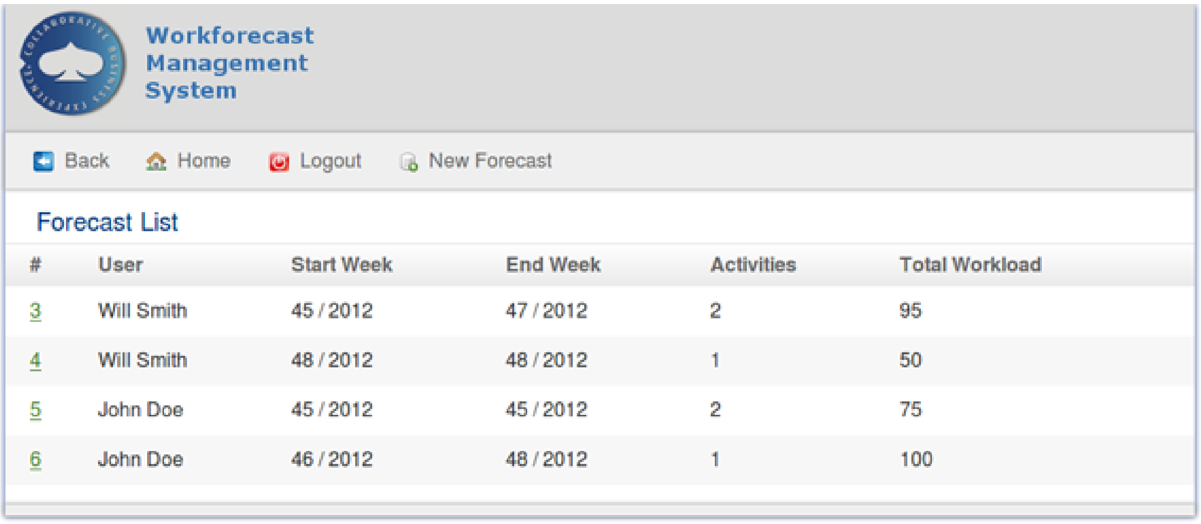
Show ActivityType

Name	Billable T&M
Paid	True
Engagement	True
Available	True

Figure 13: Activity Type show screen

5.3.4 Forecast Section

User can create forecasts in the forecast section. System checks if consultant has created forecasts when he/she clicks forecast section button in the home screen. Latest forecast is opened if there is any forecasts, otherwise create forecast screen is opened. Manager is redirected straight to the forecast list. Consultant sees only his own forecasts in the list while manager sees all forecasts.

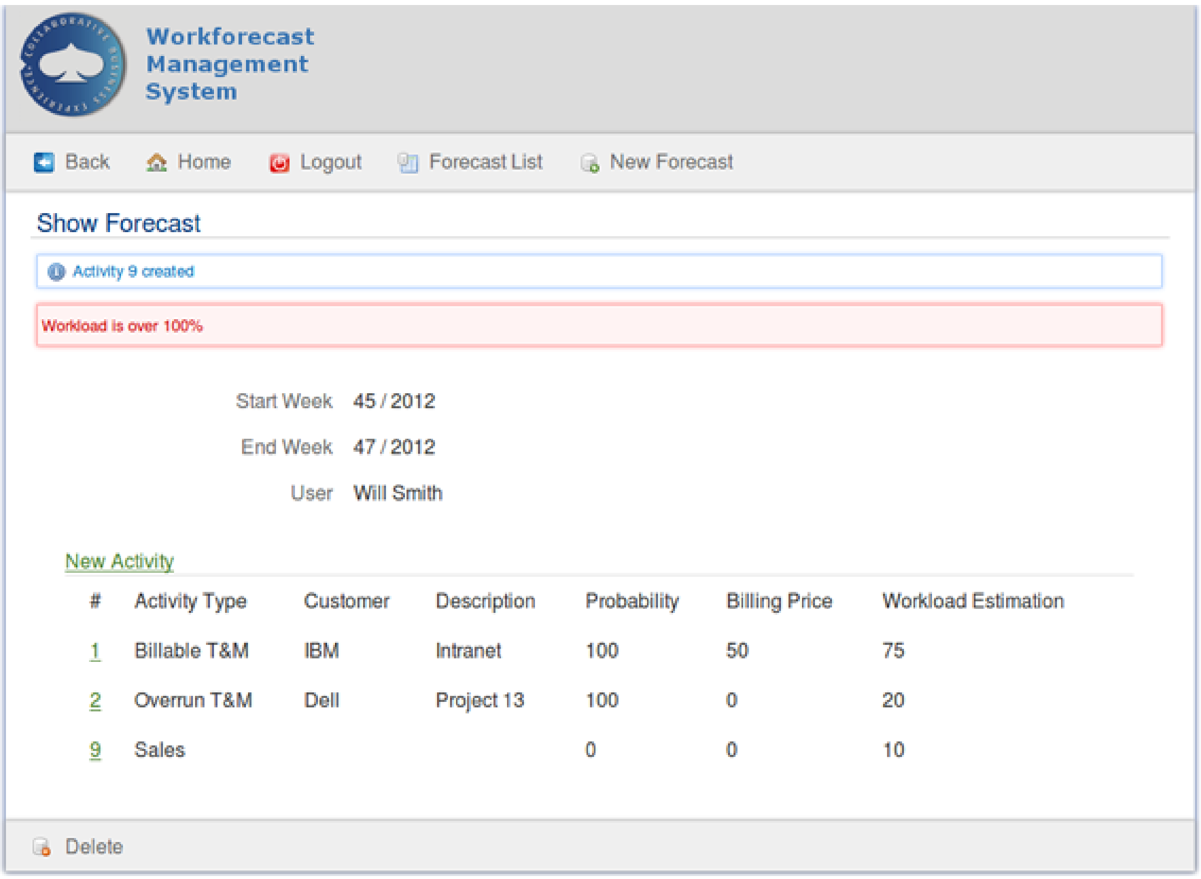


Forecast List

#	User	Start Week	End Week	Activities	Total Workload
3	Will Smith	45 / 2012	47 / 2012	2	95
4	Will Smith	48 / 2012	48 / 2012	1	50
5	John Doe	45 / 2012	45 / 2012	2	75
6	John Doe	46 / 2012	48 / 2012	1	100

Figure 14: Managers view of forecast list

Ability to add new activities is shown after the forecast is created. Added activities are shown in the forecast in table format. Activities can be added, edited and removed only from the forecast. All activities from the previous forecast are copied to a new forecast when the new forecast is created. Forecast time interval is automatically set to next week after the latest forecast if there is a latest forecast. In the case of the first forecast for the user new forecast is created. There will be an alert if the combined workload from all activities in the forecast exceeds 100%.



Show Forecast

Activity 9 created

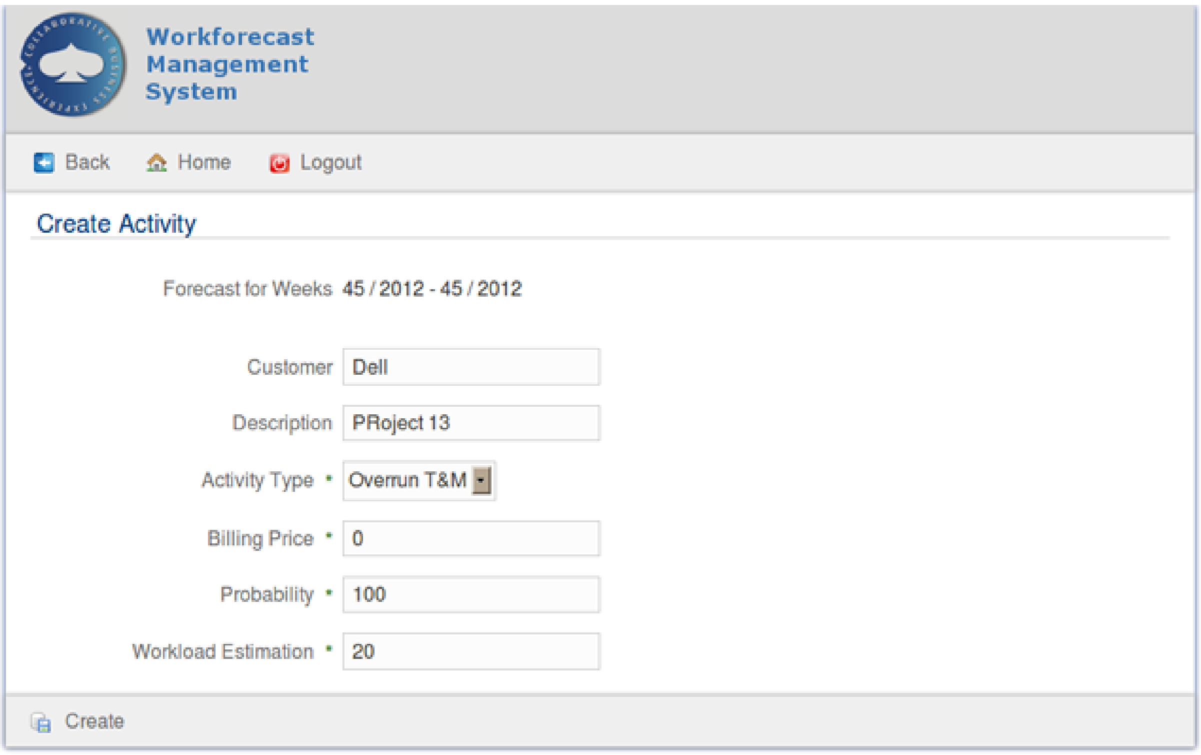
Workload is over 100%

Start Week	45 / 2012
End Week	47 / 2012
User	Will Smith

New Activity

#	Activity Type	Customer	Description	Probability	Billing Price	Workload Estimation
1	Billable T&M	IBM	Intranet	100	50	75
2	Overrun T&M	Dell	Project 13	100	0	20
9	Sales			0	0	10

Figure 15: Forecast show screen shown if workload for one week is over 100%



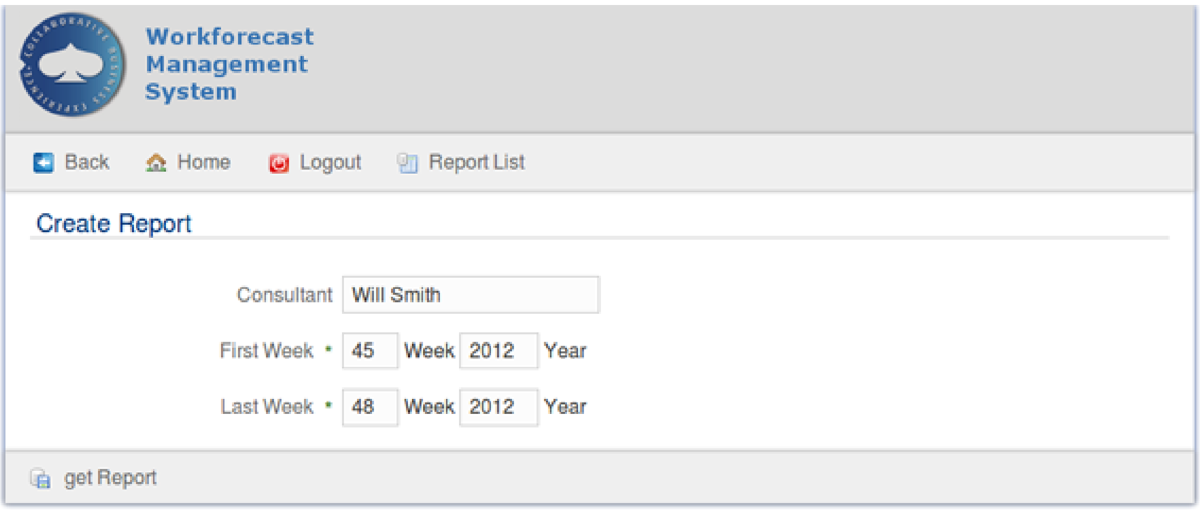
Forecast for Weeks	45 / 2012 - 45 / 2012
Customer	Dell
Description	PRoject 13
Activity Type *	Overrun T&M
Billing Price *	0
Probability *	100
Workload Estimation *	20

Figure 16: Adding activity to the forecast

5.3.5 Report Section

Available workforce can be found using reports section. Reports are created from the forecasts data. Manager or Director who is creating a report can select a consultant and a time interval for the report. Workload is calculated from all forecast which are fully or partly inside the time interval. ARVE and URVE are calculated with the help of the dynamic activity types, no fixed parameters are needed for calculations. Report shows workload total and availability for every week in the time interval. All reports are saved for later use and can be browsed in the report list screen.

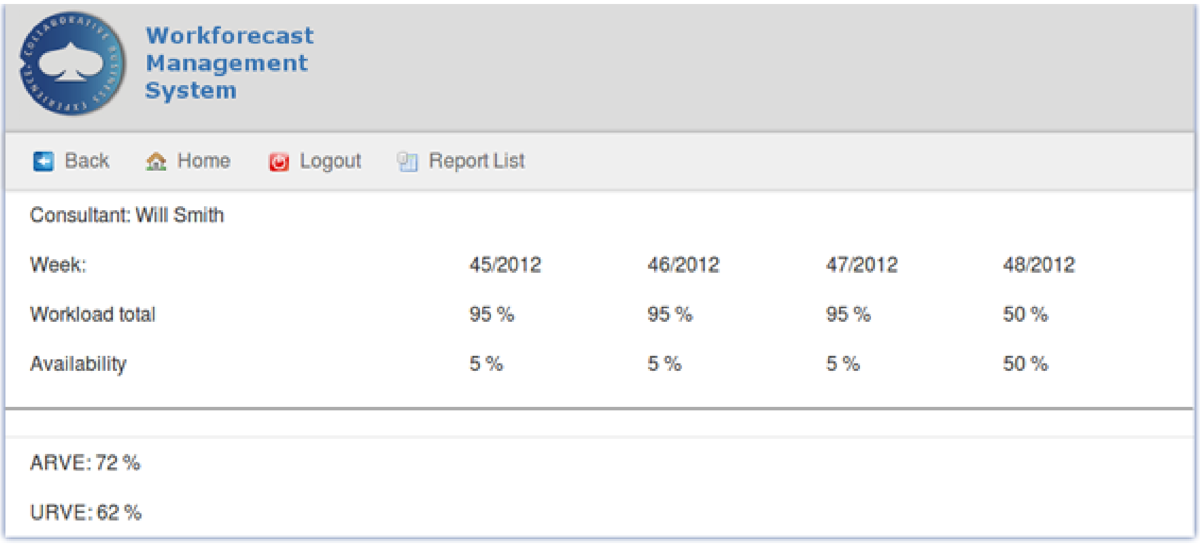
Future versions will include report menu screen for handling reports. Team based reports are planned to be implemented. There is also plans to make report creations using special queries, so manager can search for consultants which have high availability.



Create Report

Consultant	Will Smith			
First Week *	45	Week	2012	Year
Last Week *	48	Week	2012	Year

Figure 17: Creating new report



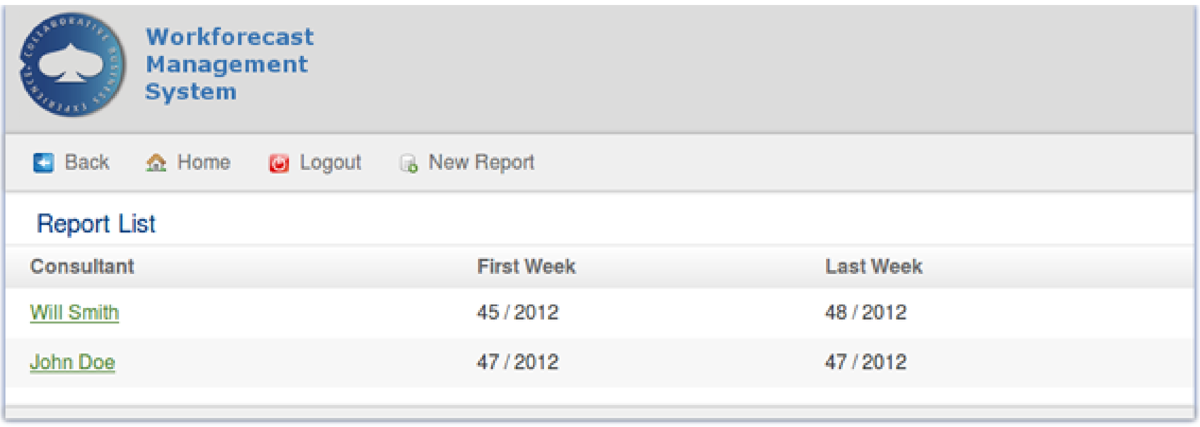
Consultant: Will Smith

Week:	45/2012	46/2012	47/2012	48/2012
Workload total	95 %	95 %	95 %	50 %
Availability	5 %	5 %	5 %	50 %

ARVE: 72 %

URVE: 62 %

Figure 18: Report layout



Report List

Consultant	First Week	Last Week
Will Smith	45 / 2012	48 / 2012
John Doe	47 / 2012	47 / 2012

Figure 19: Report list

6 Summary and Lessons Learned

The agile development code camp 2012 by Capgemini was an excellent activity to get an insight in the agile software development. For all of us the MVC pattern with GRAILS was new and we had to get used to it on our own. It is important to face a problem and to be able to find a solution for it. We think that is an important lesson learned from the code camp for almost all participants. Further we were able to improve our skills in java programming and Web-technologies like HTML and CSS. In the end we worked hard to implement as much functionality as possible. The core functionality is completely done, just some additional features are still in progress and can not be finished within the code camp.

At last but not least we are all student in the field of Software Engineering and tasks like this code camp will probably be our future life and the most important lesson learned is that the development of a new system from the very beginning is fun and we have chosen the right field of study.