Betting System
Process report

Group 10, SEP2Z-S16
Atanas Latinov (239 841)
Daniel Hamarik (239 857)
Stefan Alexiev (240 278)

Supervisors
Henrik Kronborg Pedersen
Jan Munch Pedersen

Table of Contents

Table of figures

1. Group policy

After initial discussions the group agreed to the following obligations and considerations:

All members created and decided to follow these rules:

1) I agree to meet up on a regular basis.
2) The time and place of meetings shall be agreed on by all group members.
3) If I am not able to come to a group meeting, I will contact other group members.
4) Stories are planned before their inclusion and execution in sprints.
5) We will spread the work equally between everybody.
6) I have to finish the work I am allotted to do on time and upload it to Dropbox.
7) I will communicate with others only in English.
8) If I get an idea how to do something differently, I will present it to the others.
9) I will be an active member of the group in all aspects.
10) I will write my code very clearly and understandably for others.
11) If I do not understand something, I will ask other members for help.
12) I will help others with problems in assignments.
13) I will understand the parts of the code that the other team members wrote.
14) I have the freedom to work on a task of my choice.
15) I will prepare a small revision of what I did and present it to the others during daily stand-ups.
16) If I am not satisfied with something, I will share it with my team members and work towards a solution together.

2. Working Process

The initial stage of every project that can be called successful, is the creation of a plan that separates the work in stages, hence providing the possibility for agile development. By following the process framework for agile software development Unified Process, we adopted a flexible strategy that provides us with four clearly distinguished phases in which we can focus on every aspect of the system we are building. For managing the software development part, we are using the SCRUM Framework. In combination, these two strategies provide a well-organized scheme that enable us to push every facet of the project while doing time boxed iterations.

During the *Inception* phase we outlined the goals of the system. In the its course, the team discussed and agreed upon the main features that the program should provide as well as the delimitations and functionalities that the team knew that could not be implemented for the given timeframe. On a technical level, we used a class diagram that would give us an overview of the classes, how they are connected as well as the scope of the system as a whole.

After determining the main activities that the user could do, we included them in a Use case diagram, which contained a detailed description of every main action the user could go through. The following image is the use case diagram and a use case description for the first action:

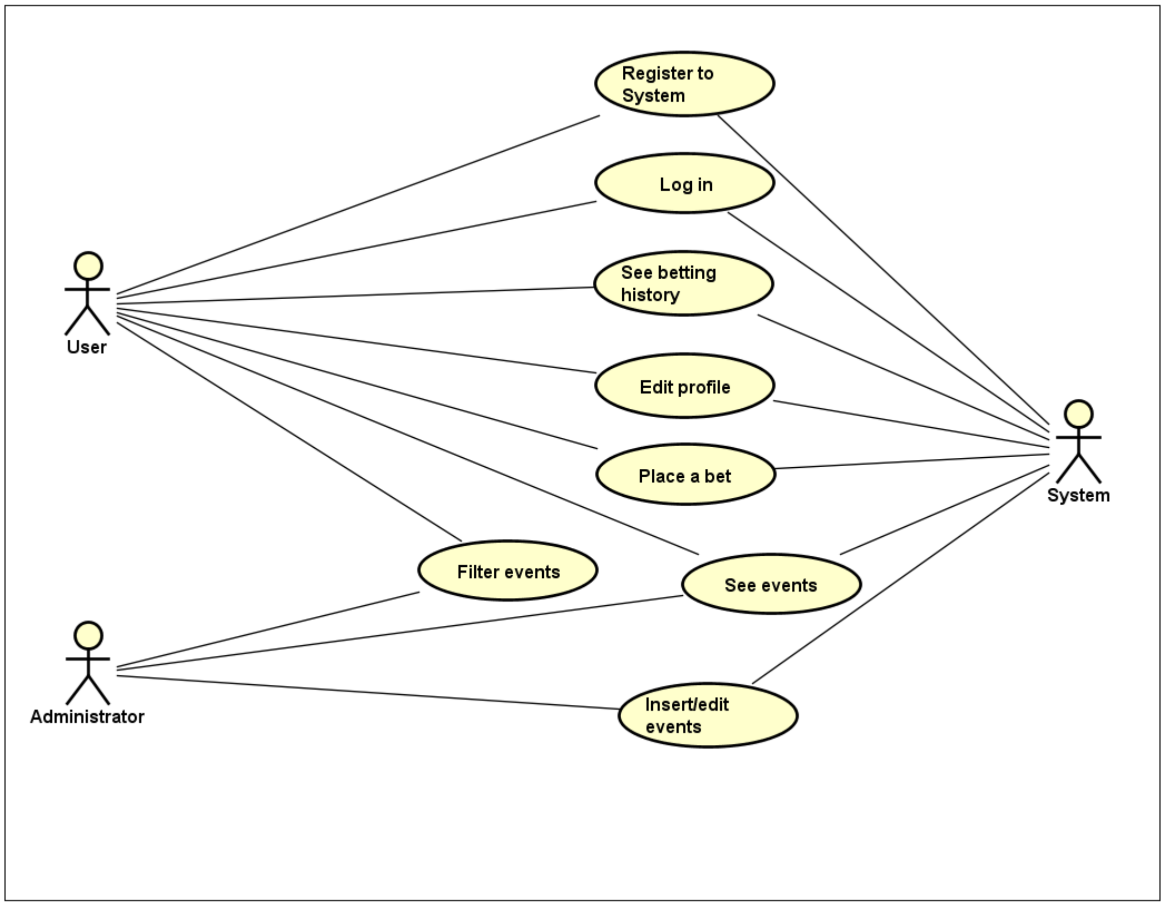
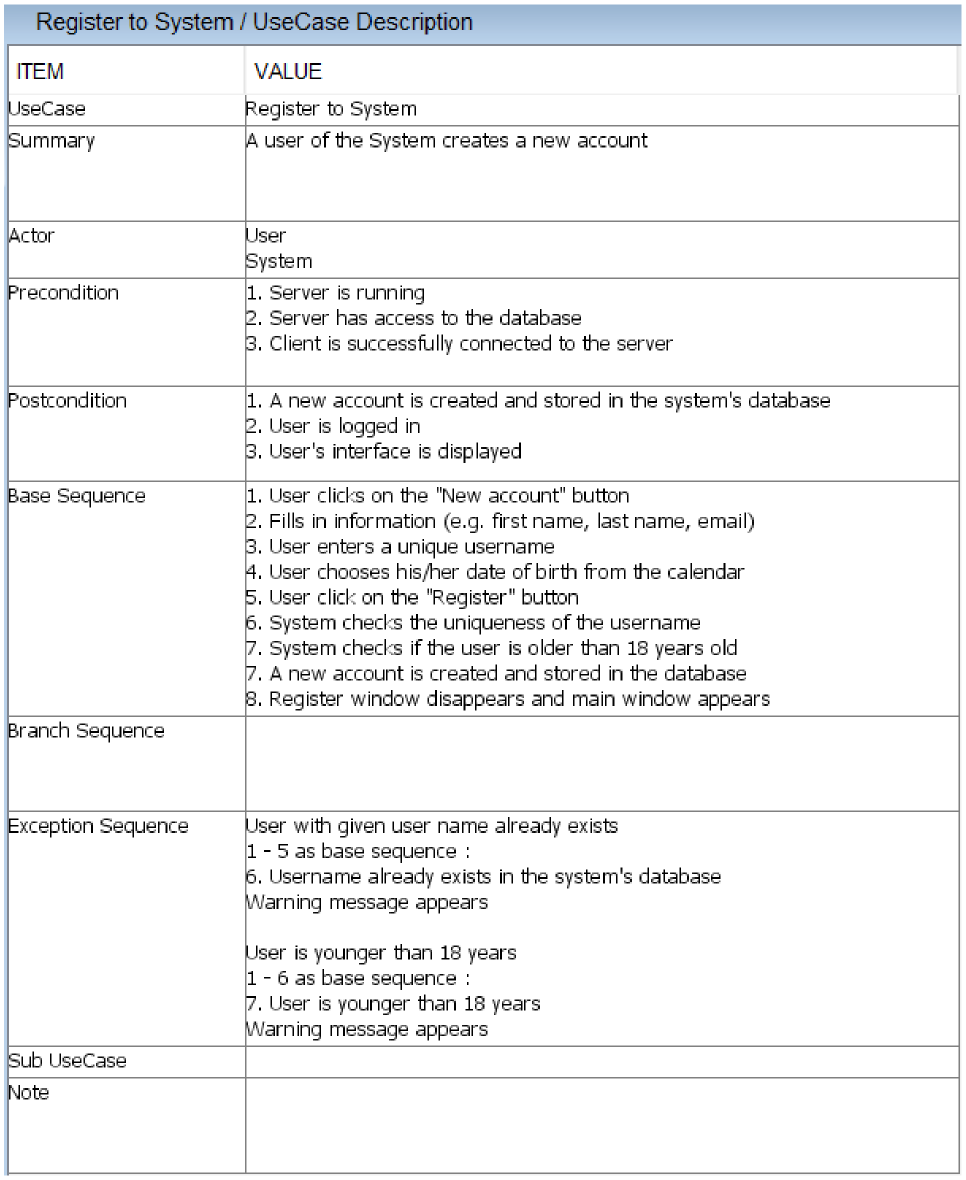


Figure 1. Betting system use case


Register to System / UseCase Description

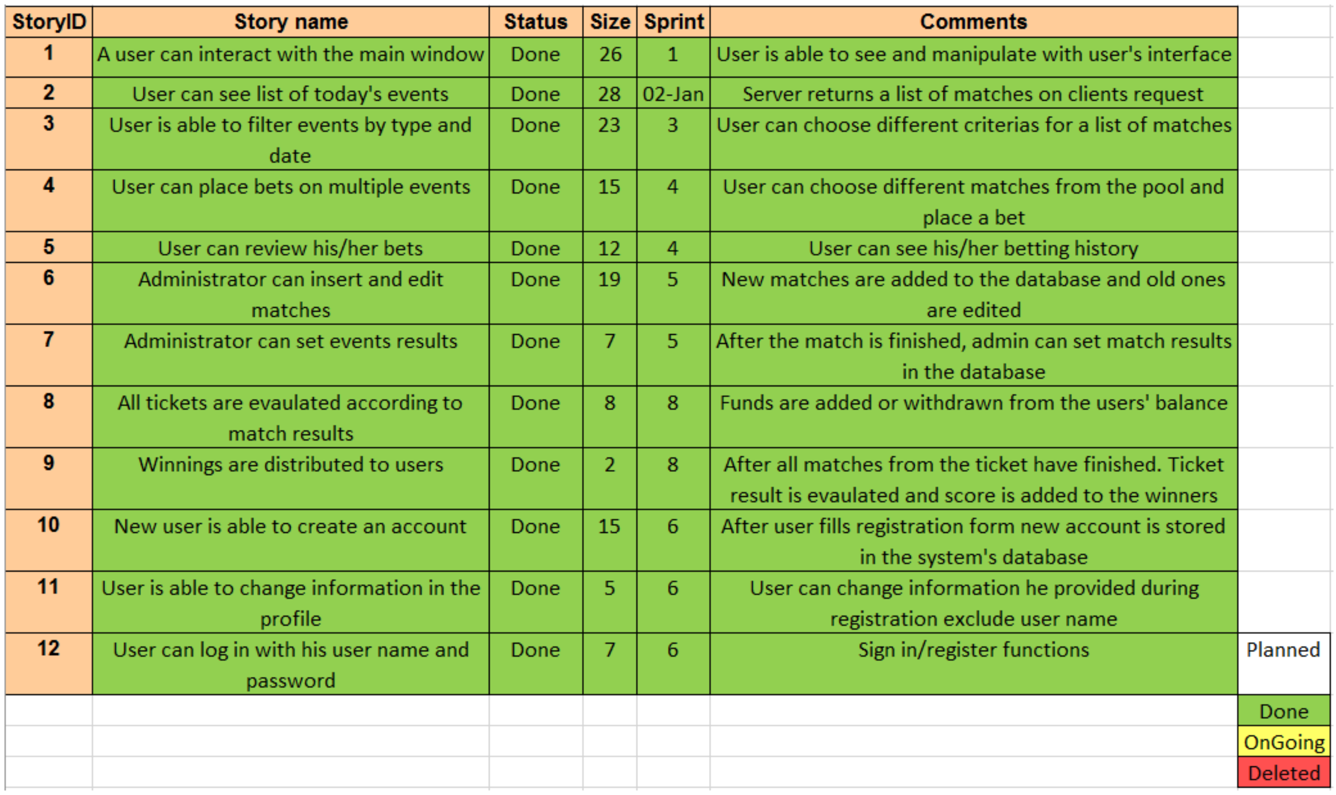
ITEM	VALUE
UseCase	Register to System
Summary	A user of the System creates a new account
Actor	User System
Precondition	1. Server is running 2. Server has access to the database 3. Client is successfully connected to the server
Postcondition	1. A new account is created and stored in the system's database 2. User is logged in 3. User's interface is displayed
Base Sequence	1. User clicks on the "New account" button 2. Fills in information (e.g. first name, last name, email) 3. User enters a unique username 4. User chooses his/her date of birth from the calendar 5. User click on the "Register" button 6. System checks the uniqueness of the username 7. System checks if the user is older than 18 years old 7. A new account is created and stored in the database 8. Register window disappears and main window appears
Branch Sequence	
Exception Sequence	User with given user name already exists 1 - 5 as base sequence : 6. Username already exists in the system's database Warning message appears User is younger than 18 years 1 - 6 as base sequence : 7. User is younger than 18 years Warning message appears
Sub UseCase	
Note	

Figure 2. Register to system description

Another very useful tool for clarification of the possible action sequences is the activity diagram. We created such diagrams for every sequence of activities that the system provides. During the course of development, they turned out to be a great reference material, especially for the user interface design, since they helped us visualize the connection between the frames and other components.

After we outlined the main functions of the system, we moved on to the next UP phase – *Elaboration*. In the initial steps of this stage we started creating our product backlog. This main SCRUM artefact proved to be extremely useful as it was a reference list of all the functionality, desired in the product. This was the predominant way for us as a team to identify features in the form of user stories, which are short, simple descriptions of the desired functionality told from our perspective. The backlog also helped us identify the main risks from a programming and design standpoint. In our case, those proved to be the creation of the relational database, as well as its integration within our system. Another aspect, that we considered to be a challenge, was the Client-Server connection and the necessary architectural changes we had to consider.

What followed after identifying the larger tasks and prioritizing them in the list was to determine what had to be done in order for them to be implemented and fully functioning. During this process, we used the activity diagrams, coupled with the use case diagram and the use case diagram descriptions, as reference points from which we extracted the smaller tasks that had to be accomplished. The image below presents the product backlog. We would also like to acknowledge the fact that the backlog underwent several changes during the development process. This is due to the agile nature of the software development frameworks and their strategies that we have adopted for the current project.



StoryID	Story name	Status	Size	Sprint	Comments	
1	A user can interact with the main window	Done	26	1	User is able to see and manipulate with user's interface	
2	User can see list of today's events	Done	28	02-Jan	Server returns a list of matches on clients request	
3	User is able to filter events by type and date	Done	23	3	User can choose different criterias for a list of matches	
4	User can place bets on multiple events	Done	15	4	User can choose different matches from the pool and place a bet	
5	User can review his/her bets	Done	12	4	User can see his/her betting history	
6	Administrator can insert and edit matches	Done	19	5	New matches are added to the database and old ones are edited	
7	Administrator can set events results	Done	7	5	After the match is finished, admin can set match results in the database	
8	All tickets are evaulated according to match results	Done	8	8	Funds are added or withdrawn from the users' balance	
9	Winnings are distributed to users	Done	2	8	After all matches from the ticket have finished. Ticket result is evaulated and score is added to the winners	
10	New user is able to create an account	Done	15	6	After user fills registration form new account is stored in the system's database	
11	User is able to change information in the profile	Done	5	6	User can change information he provided during registration exclude user name	
12	User can log in with his user name and password	Done	7	6	Sign in/register functions	Planned
						Done
						OnGoing
						Deleted

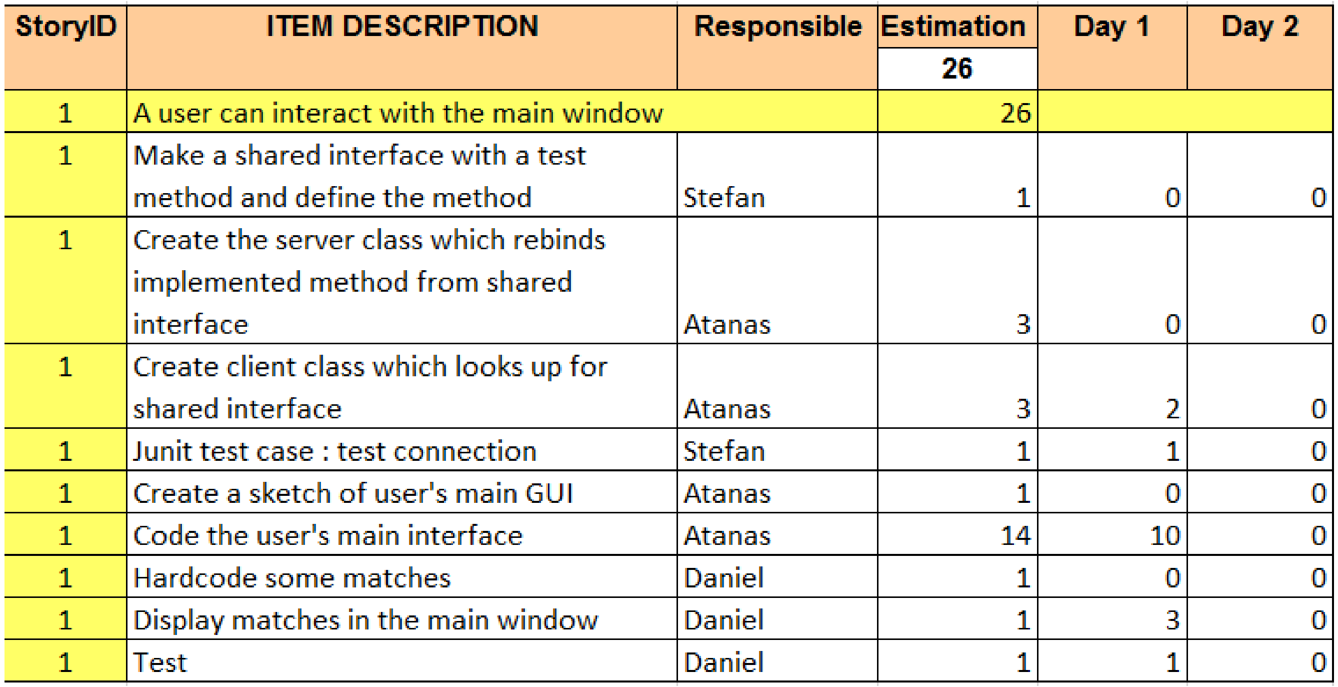
Figure 3. Final version of the product backlog

The next step of the development process, according to the SCRUM framework, was to determine the number of sprints as well as the size of each task in it and the total size of all the tasks. As can be seen from the image above, we ended up with 12 stories with each one having a number of several sub stories. In the following section, the reader can get familiar with the sprints and their assigned tasks.

2.1 Sprints

There were several factors that influenced our decision regarding the number of sprints and their individual sizes. One of the more important ones was the time limitation. We had to make an approximation for the size of each sprint so that it would be possible for all of them to be finished in time, even if unexpected errors occur and stories would have to be revisited and updated. After taking everything into account we ended up with 7 sprints and an average sprint size of 25 hours, well within the time frame specified in the project description. It is also important to note that each sprint is spread across two days, which makes the total time allotted for sprints equal to 14 days.

The main focus of the first sprint is the creation of a Client-Server connection and the main user interface – elements that lay the foundations of the program. The picture below shows the various sub stories and presents the reader with information about who is responsible for each story and what is its estimated size.



StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			26		
1	A user can interact with the main window		26		
1	Make a shared interface with a test method and define the method	Stefan	1	0	0
1	Create the server class which rebinds implemented method from shared interface	Atanas	3	0	0
1	Create client class which looks up for shared interface	Atanas	3	2	0
1	Junit test case : test connection	Stefan	1	1	0
1	Create a sketch of user's main GUI	Atanas	1	0	0
1	Code the user's main interface	Atanas	14	10	0
1	Hardcode some matches	Daniel	1	0	0
1	Display matches in the main window	Daniel	1	3	0
1	Test	Daniel	1	1	0

Figure 4. Sprint 1

The following sprint contains stories that continue to lay the foundations of the program. The main focus is the creation of a normalized relational database. Furthermore, connection to this database and Java is established in this sprint with defined methods for data extraction and insertion. Basically here we defined the bridge between the database and the java code. Apart from the database, the cardinal classes are also created in this sprint.

StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			28		
2	User can see list of today's events		28		
2	Collect all data which has to be stored in the database	Atanas	1	0	0
2	Develop a relational database model using normalization desing technique	Atanas	3	0	0
2	Document normalization process	Atanas	1	1	0
2	Test database functional dependencies and update anomalies	Atanas	1	0	0
2	Create database in postgreSQL	Atanas	1	1	0
2	Test : populate tables	Atanas	1	1	0
2	Connect java to the Database	Daniel	2	0	0
2	Define update method	Daniel	1	0	0
2	Define a SELECT method	Daniel	1	1	0
2	Test connection, insert and update methods	Daniel	1	1	0
2	Junit test case	Daniel	1	1	0
2	Class Data which contains information about user based on database	Stefan	1	0	0
2	Create class User with username, password and an instance of the Data class	Stefan	1	0	0
2	Class Match with all information needed	Stefan	2	1	0
2	Create class Bet with information about a single bet	Stefan	1	1	0
2	Create class Ticket which contains information about a user and an array list of bets	Stefan	2	2	0
2	Design how to send a list of matches over a network	Atanas	1	1	
2	Create a method which takes matches from database and displays them in the JTable	Atanas	5	1	0
2	Test: display matches from the database in a JTable	Atanas	1	1	0

Figure 5. Sprint 2

The target in sprint 3 is the filtering function and what has to be implemented for it to be operational.

StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			23		
3	User can filter events by type/date		23		
3	Code the query method which selects matches based on their type	Daniel	3	0	0
3	Test query	Daniel	1	0	0
3	Query method which selects matches in a given time period	Daniel	3	1	0
3	Validating method for wrong time periods	Atanas	3	0	0
3	Test select methods	Daniel	1	1	0
3	Design Controller class for the main GUI	Atanas	2	0	0
3	Add an actionListener to "Filter" button	Atanas	4	2	0
3	Connect queries with "Filter" button	Atanas	4	4	0
3	Test: check results in JTable	Stefan	2	2	0

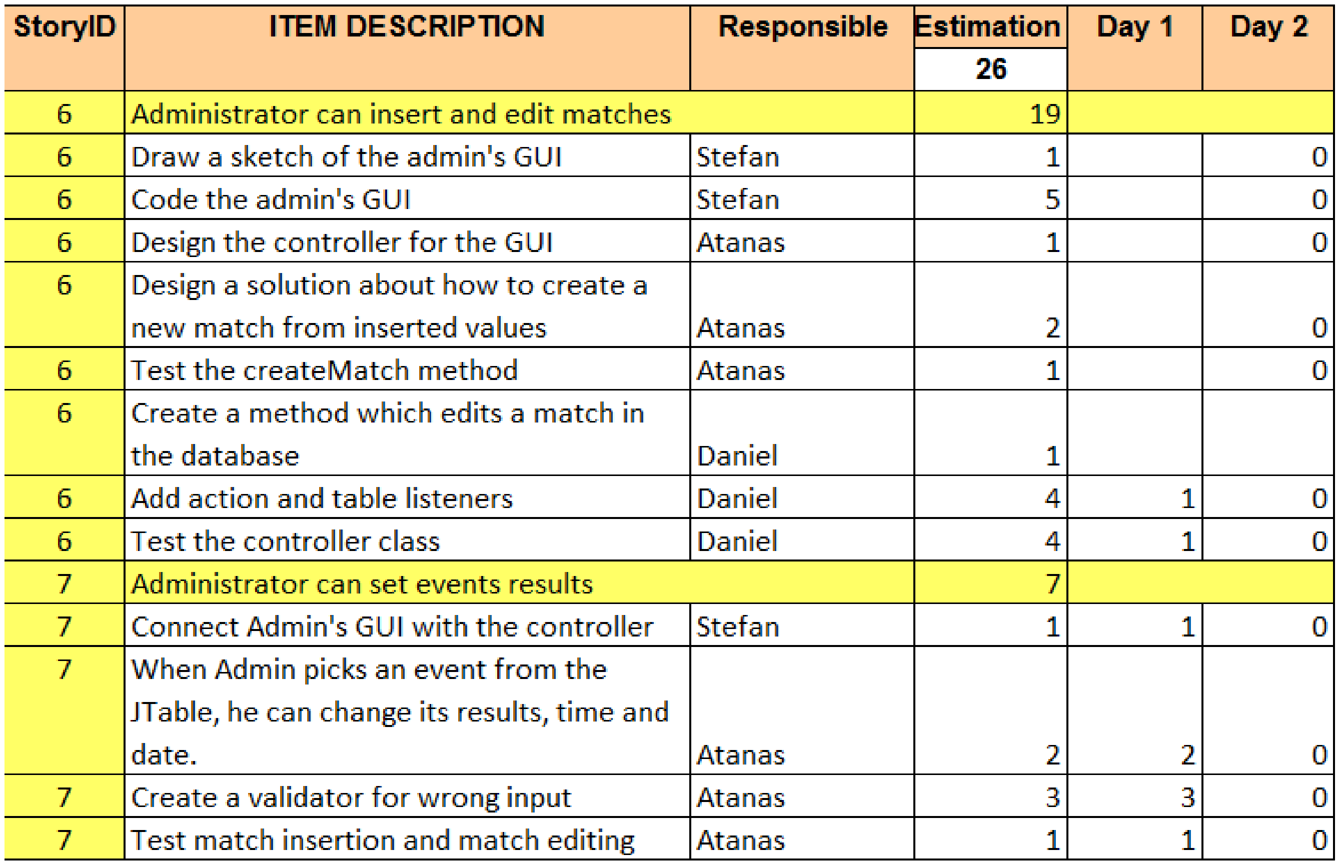
Figure 6. Sprint 3

Sprint 4 focuses on the main feature of the system that is placing a bet as well as reviewing your betting history. The big challenge during this sprint was to figure out how the user can add matches as well as make winner predictions for the same matches at the same time. After some discussions and different ideas about the design, we all agreed on the double clicking feature (for more information about the feature check the project report) which was approved by all team members and eventually included in the system.

StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			27		
4	User can place bets on multiple events		15		
4	Design and code virtual ticket GUI	Stefan	4	1	0
4	Add a mouseListener to the JTable	Atanas	3	0	0
4	On double click, add a row to the JTable in the virtual ticket	Atanas	1	1	0
4	Automate adding the coefficient based on a selected column	Atanas	1	1	0
4	Add a button ActionListener in controller	Atanas	1	1	0
4	Create a ticket from selected matches	Atanas	2	2	0
4	Test created ticket	Atanas	1	1	0
4	Code the method which inserts a ticket in the database	Atanas	1	1	0
4	Test ticket insertion	Atanas	1	1	0
5	User can review his/her bets		12		
5	Design and code the history interface	Daniel	4	0	0
5	Test: display String values in a history panel	Daniel	1	0	0
5	All tickets are displayed in the history tabbed pane	Daniel	1	0	0
5	On ticket selection, the JTable is filled with detailed information about the selected ticket	Daniel	2	2	0
5	Test displaying of a ticket	Daniel	1	1	
5	Test displaying of ticket details	Daniel	1	1	0
5	Figure out how to refresh tickets and results	Daniel	1	3	0
5	Test refreshing	Daniel	1	1	0

Figure 7. Sprint 4

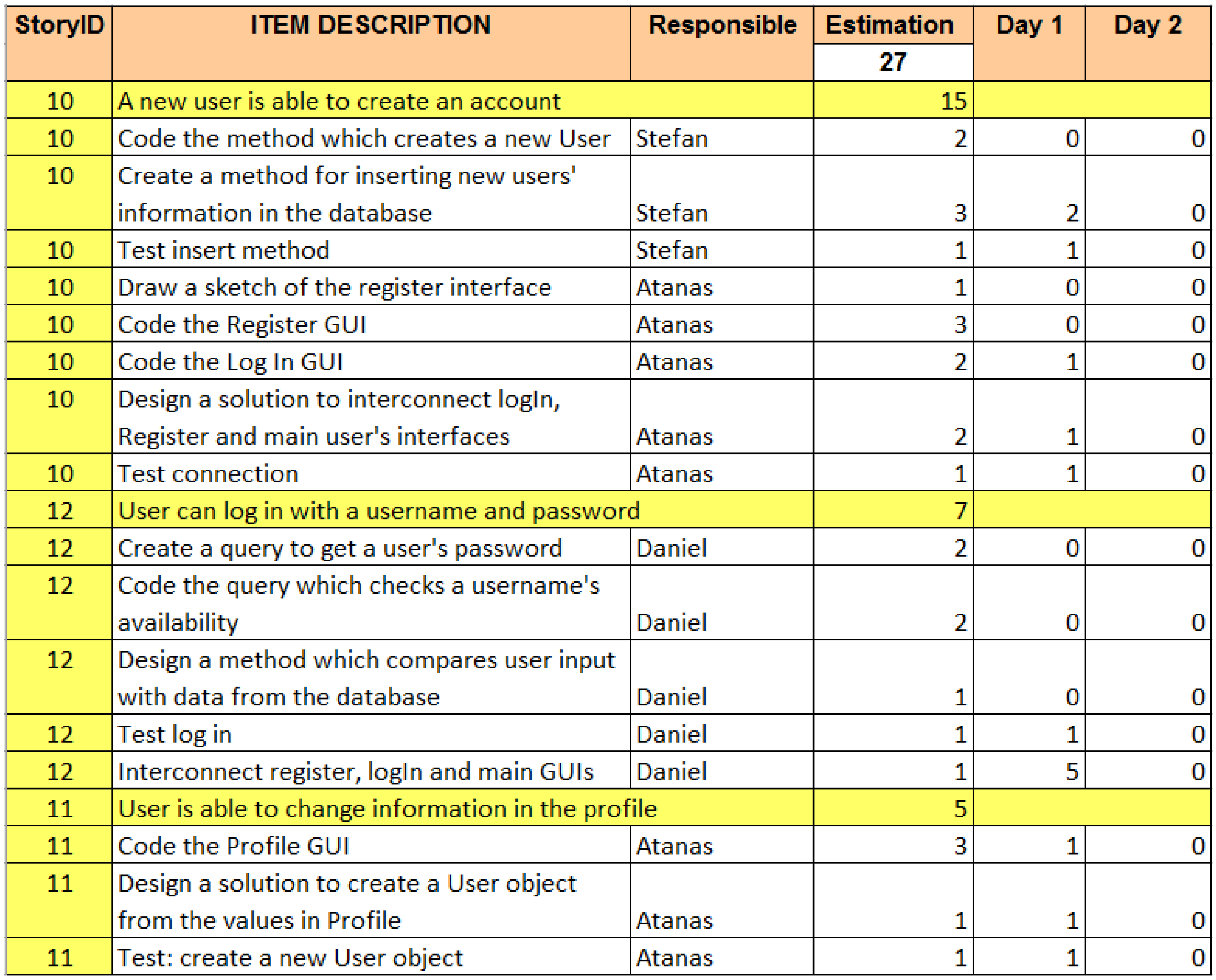
In sprint 5, the team decided to put their efforts in developing the necessary functions and interfaces for the administrator side of the system. The interesting thing with the administrator is that he has an entirely different user interface, which means that an entirely different MVC module should be created for it. Another challenge from a design standpoint was how we could include the whole functionality for the admin in one frame without having to transfer the user to different windows and frames.



StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			26		
6	Administrator can insert and edit matches		19		
6	Draw a sketch of the admin's GUI	Stefan	1		0
6	Code the admin's GUI	Stefan	5		0
6	Design the controller for the GUI	Atanas	1		0
6	Design a solution about how to create a new match from inserted values	Atanas	2		0
6	Test the createMatch method	Atanas	1		0
6	Create a method which edits a match in the database	Daniel	1		
6	Add action and table listeners	Daniel	4	1	0
6	Test the controller class	Daniel	4	1	0
7	Administrator can set events results		7		
7	Connect Admin's GUI with the controller	Stefan	1	1	0
7	When Admin picks an event from the JTable, he can change its results, time and date.	Atanas	2	2	0
7	Create a validator for wrong input	Atanas	3	3	0
7	Test match insertion and match editing	Atanas	1	1	0

Figure 8. Sprint 5

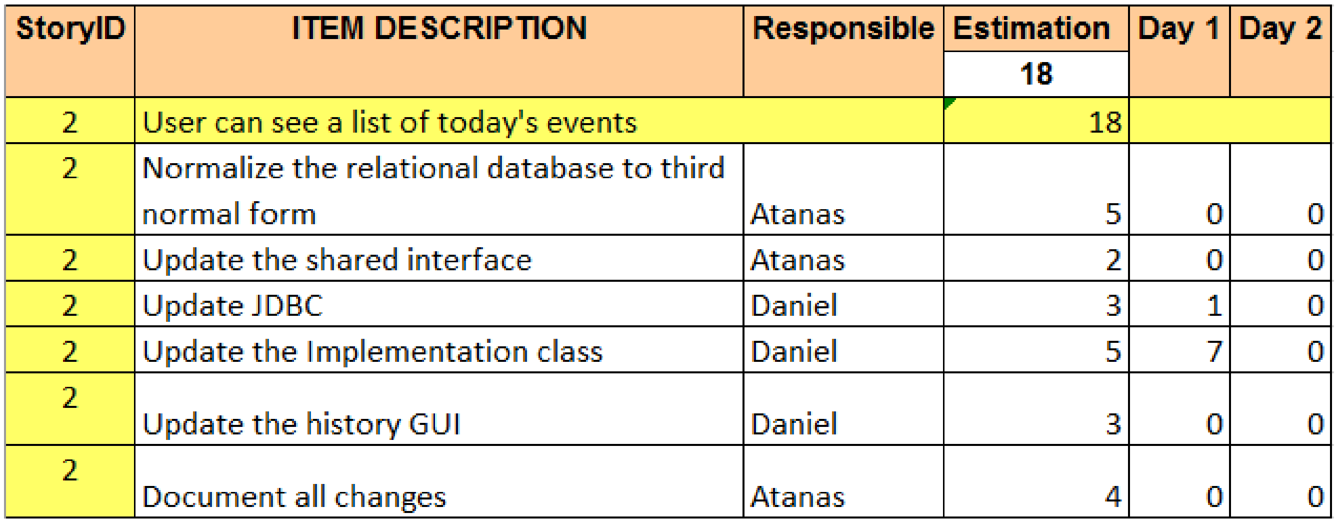
Sprint 6 focuses on the user authentication process and the interaction with the database. There are also several stories, regarding the user interface, where the user enters the credentials necessary for login or registering.



StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			27		
10	A new user is able to create an account		15		
10	Code the method which creates a new User	Stefan	2	0	0
10	Create a method for inserting new users' information in the database	Stefan	3	2	0
10	Test insert method	Stefan	1	1	0
10	Draw a sketch of the register interface	Atanas	1	0	0
10	Code the Register GUI	Atanas	3	0	0
10	Code the Log In GUI	Atanas	2	1	0
10	Design a solution to interconnect logIn, Register and main user's interfaces	Atanas	2	1	0
10	Test connection	Atanas	1	1	0
12	User can log in with a username and password		7		
12	Create a query to get a user's password	Daniel	2	0	0
12	Code the query which checks a username's availability	Daniel	2	0	0
12	Design a method which compares user input with data from the database	Daniel	1	0	0
12	Test log in	Daniel	1	1	0
12	Interconnect register, logIn and main GUIs	Daniel	1	5	0
11	User is able to change information in the profile		5		
11	Code the Profile GUI	Atanas	3	1	0
11	Design a solution to create a User object from the values in Profile	Atanas	1	1	0
11	Test: create a new User object	Atanas	1	1	0

Figure 9. Sprint 6

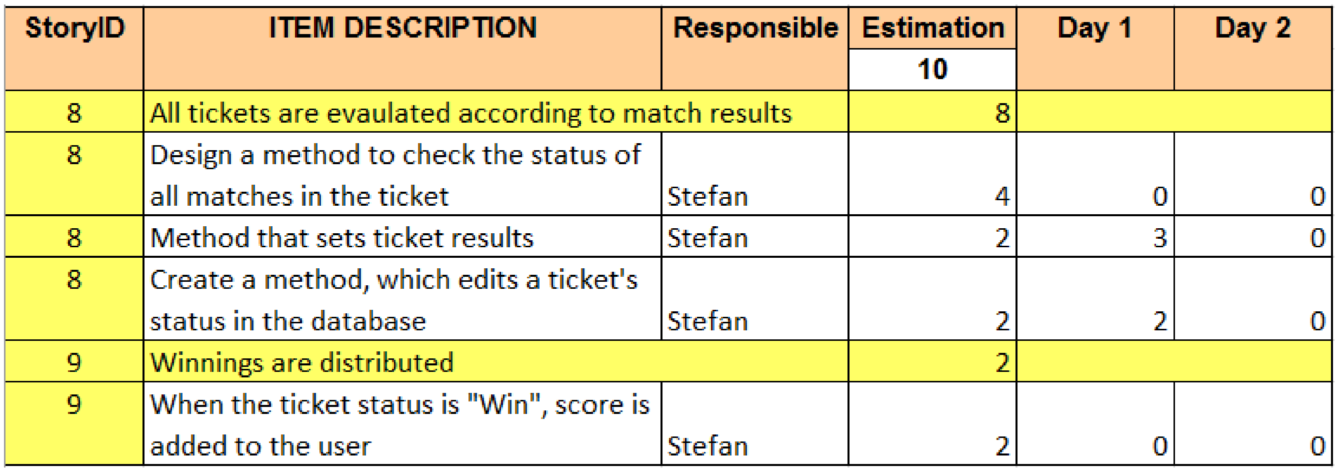
The interesting thing about sprint 7 is that it was created from a meeting with the project's supervisor. After the database, that was used at the time, was carefully inspected, we were advised to modify it. After estimations of the amount of work that has to be invested into it, we decided to create a new sprint with a size of 18 that would cover all the necessary changes for adapting the modified database with the code and vice versa.



StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			18		
2	User can see a list of today's events		18		
2	Normalize the relational database to third normal form	Atanas	5	0	0
2	Update the shared interface	Atanas	2	0	0
2	Update JDBC	Daniel	3	1	0
2	Update the Implementation class	Daniel	5	7	0
2	Update the history GUI	Daniel	3	0	0
2	Document all changes	Atanas	4	0	0

Figure 10. Sprint 7

The final sprint deals with ticket evaluation and the distribution of the winnings. The challenge for the team here is the algorithm for ticket status checks, and for the distribution of the winnings depending on the results.

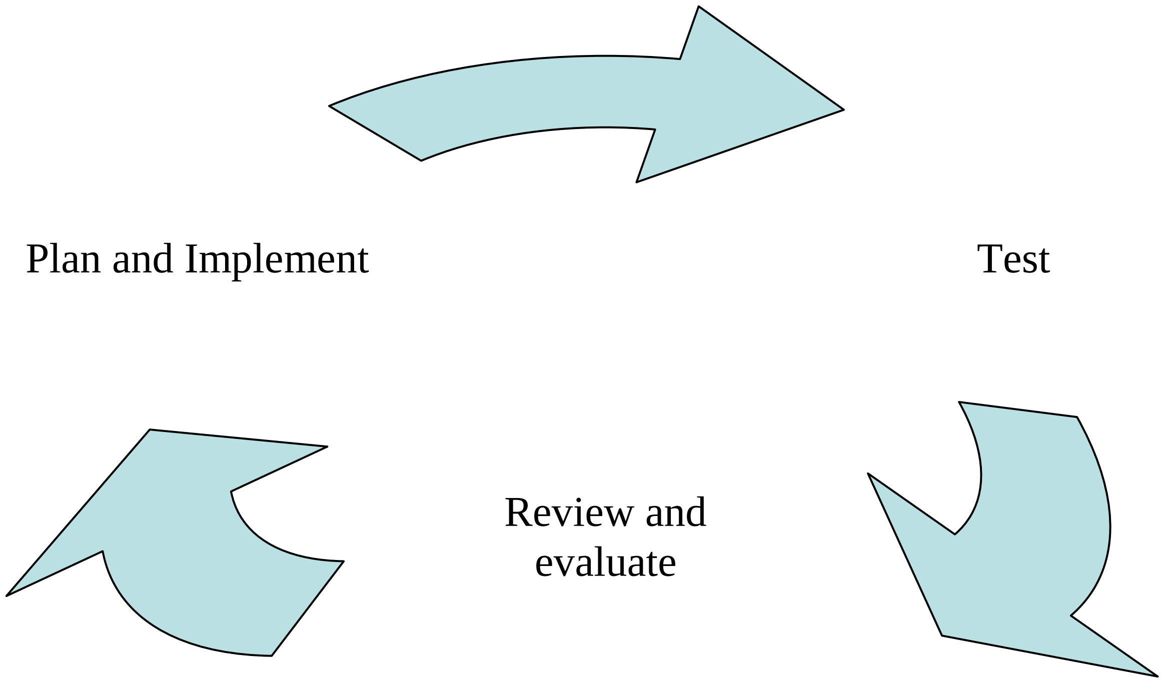


StoryID	ITEM DESCRIPTION	Responsible	Estimation	Day 1	Day 2
			10		
8	All tickets are evaulated according to match results		8		
8	Design a method to check the status of all matches in the ticket	Stefan	4	0	0
8	Method that sets ticket results	Stefan	2	3	0
8	Create a method, which edits a ticket's status in the database	Stefan	2	2	0
9	Winnings are distributed		2		
9	When the ticket status is "Win", score is added to the user	Stefan	2	0	0

Figure 11. Sprint 8

This concludes the Sprints section.

With the start of the *Construction* phase, it was paramount for us to have an executable software module that would mark our progress after each sprint and give room for improvement or a complete revamp of a given feature, if necessary. After the end of each iteration, the team tested the newly created module as well as documented and reported any undesirable activities or inefficiencies of the software, so bugs and problems would be addressed on the spot rather than being left for fixing at a later stage, which would additionally overcomplicate the adjustment process later. Here it is important to note that Junit tests were created for the core functionalities, whereas the tests from a user standpoint occurred after we had developed the necessary user interfaces.



In the course of the Transition phase, we took the roles of target users. After going through the several use cases over and over, we exchanged what results we had from the tests and discussed possible refinements that could be incorporated over the course of several Transition phase iterations. It is important to note that during this phase, after a meeting with the project's supervisor, we established that a major database overhaul had to be done. Of course, the amount of work needed for this remodeling meant that we had to make another sprint only dedicated to the database change as well as its adaptation to the code. After that, we repeated our working cycle for the iteration and concluded the work in the last phase.

3. Meetings

Throughout the course of the project development, we had several meetings including two meetings with a supervisor. The main goal of the meetings was to review our work as well as to share any new ideas or aspects that might need to be altered. We had several types of meetings adopted from the frameworks we were using during development:

1) Sprint planning

During those meetings, which we had before every sprint, the whole team gathered and planned each individual item in it. In these meetings we made estimations about the size of each story and about the size of the sprint as a whole.

2) Stand-ups

In the time of the stand-ups, we quickly assessed the work done and what work followed. Those meetings took place before commencing work on the sprint the team was focused on and were helpful in that we were quickly informed about the status of the daily tasks as well as the sprint.

3) Iteration reviews

Iteration reviews were paramount because after those reviews, we felt more relaxed about a given feature, given that we thoroughly reviewed it and tested it. It is important to acknowledge that whatever part of the program we ended up reviewing during those meetings, it had to be fully demonstrable which means that we are not only testing the main feature but also the other functions (sub stories) required for it to be fully operational.

4) Supervisor meetings

Supervisor meetings proved to be extremely useful and important for the proper project development in the long run. We had two supervisor meetings in total during the creation of this project. Below is a brief description of the subjects discussed:

1st meeting: At the first meeting we cleared several uncertainties related to SCRUM and UP. The product backlog underwent several changes due to the supervisor's guidance. An example of such a change is the first 5 stories in the backlog being rewritten as if they are from the user's perspective. Here we would like to note that in this meeting was acknowledged the wrong design of our relational database. After careful scrutiny on the supervisor's part, we had to alter the database and normalize it.

2nd meeting: The main theme of the second meeting was the feedback from the supervisor about the revamped database design. It took place after the added sprint, whose focus was addressing the relational database issues. The result of this meeting was that we had correctly followed the advice of the supervisor and the changes we had committed were acceptable. (For the interested reader, a comparison of the old and the current version of the relational database is available in the Appendix).

4. Project Result

As a team we believe that the SCRUM and UP frameworks were extremely beneficial and allowed us to follow a solid working strategy that does not restrict us to make changes but rather embraces them. The various meetings being held, initially considered tedious and unnecessary, actually boosted the team's efficiency and encouraged us to adopt new working ethic values with the main focus being, sharing and cooperating. Furthermore, the evolution of the software we were creating with time, especially the changes that had to be made regarding the database, has shown us that there is always room for improvement. By doing SCRUM, the rate of things being overlooked was decreased dramatically compared to previous projects. This only serves as proof of the huge beneficial effect this framework provides. Even though we had to do a major overhaul of the database, thanks to the iterative nature of our working process and the exceptional time management, we succeeded in patching it by making only one additional sprint.

The burndown chart is a precise representation of the work done for the time allotted for the project:

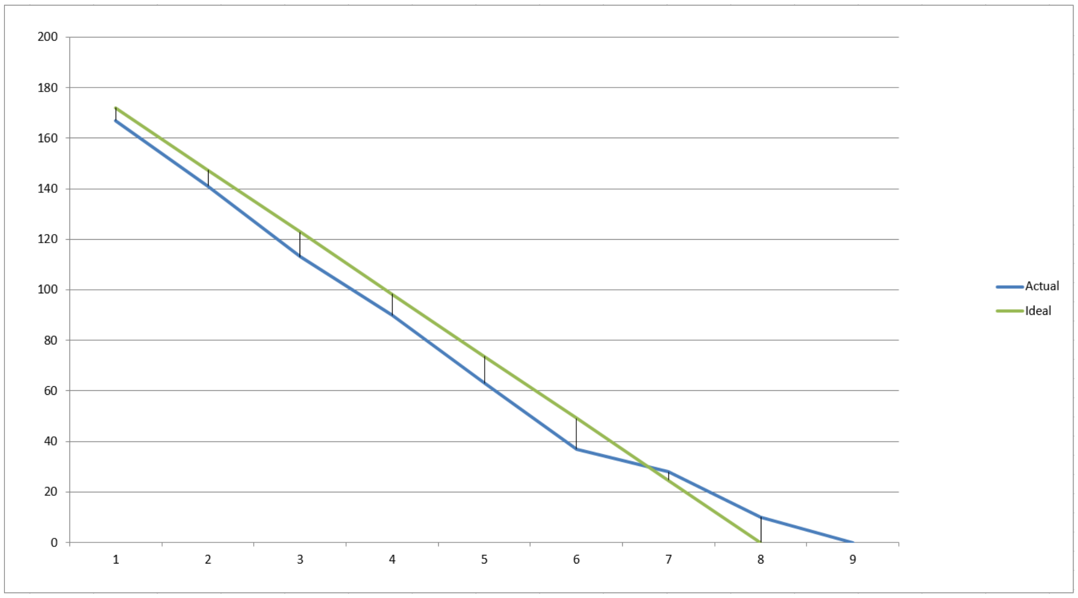
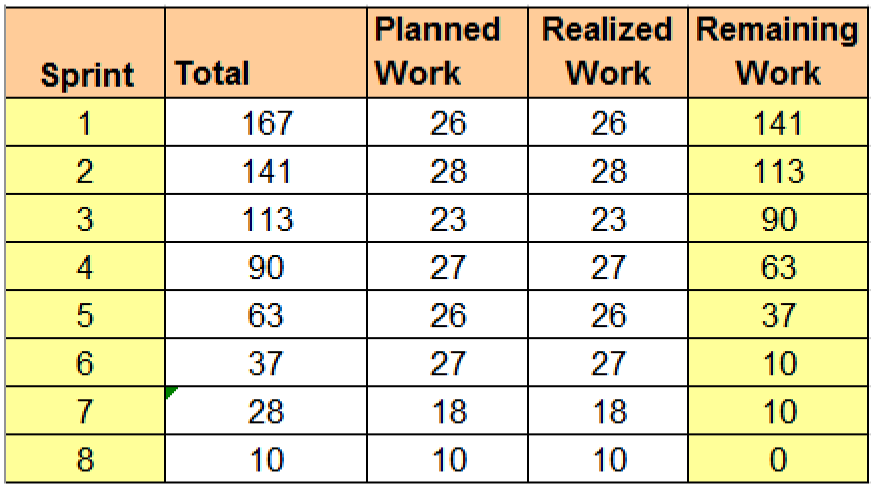


Figure 12. Betting system burndown chart



Sprint	Total	Planned Work	Realized Work	Remaining Work
1	167	26	26	141
2	141	28	28	113
3	113	23	23	90
4	90	27	27	63
5	63	26	26	37
6	37	27	27	10
7	28	18	18	10
8	10	10	10	0

Figure 13. Work size

The burndown chart shows the team's performance throughout the development process. The addition of the extra sprint and the work done related to the database remodeling, is clearly indicated by the end of the actual work remaining line.

Regarding future work improvements or things we would change, on the top of our list are the supervisor meetings and how they augment the end project result, since we experienced this effect during this task ourselves.

Another thing to consider for future projects is to properly evaluate and consider the risk factors of a project so if further problems occur, there is enough time to address them.

5. Personal reflections

Atanas Latinov:

The main requirements of this project, the first being a Client-Server system, and the second: to use a relational database, definitely added more depth to the project we had to make. A huge amount of the technologies that we used, were new to me and that made it more challenging, thus more interesting. The other engaging aspect of the project was work methodology and what strategies and methods we were going to follow in order to successfully create the product as well as satisfy the given requirements. Up until the beginning of the semester I had never heard of Unified Process or about the development framework that is SCRUM. Needless to say, I had to quickly bring myself up to speed in order to be able to enhance the entire team's effectiveness and provide help when needed. I have to say that following those strategies gave me confidence that if we as a team continue to follow the guidelines and use the tools properly, we will have no problems creating and documenting the system in the given time, even if we are three people. Eventually it turned out as expected and even though we had to make an additional sprint and address the database issues, we still managed to complete everything before the deadline. What I gained from this project is definitely the importance of defining or using a robust working strategy. Good team work is not only great synergy between the team members. If you want to build a successful project, you have to be disciplined and follow a carefully devised set of rules and guidelines and SCRUM and UP definitely provided us with all that.

Daniel Hamarik:

During the second semester, we have been learning about collections, design patterns, client-server connection, databases, SCRUM as well as unified process. I am very glad that we did this project where we had a chance to apply all the theoretical knowledge we gained during the semester. We followed SCRUM as agile software development framework. From the beginning it was a bit chaotic but after a couple of iterations, I consider it as a very useful tool in system development. As a result of using scrum artefacts and ceremonies, our meetings were well structured and everybody knew what is going on. I really enjoyed all group meetings, since all of them were highly productive. However, if I could go back in time, I would change some things, because we work agile and we are always evolving as well as the system. I would definitely change RMI because of the connection issues and instead of it, use sockets. To summarize, I am really satisfied with our Betting system.

Stefan Aleksiev:

About my personal reflection on the semester project I can say that I learnt more about how to structure a big project, from where is the best point to start and more about working as a group. I think we did good work on the whole project and I can say that almost everything which was planned, was done on time, which is really satisfying for me. I hope our next semester project will be better.