Project Name	...
Online team meeting	https://fau.zoom-x.de/j/67529253146
Production system (if any)	...
Test system (if any)	...
GitHub repository	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin
GitHub feature board	https://github.com/orgs/amosproj/projects/26/views/2
GitHub impediments backlog	https://github.com/orgs/amosproj/projects/36
Team T-shirt (white)	https://www.shirtinator.de/loadBasket/w2sVI72Xs18
Team T-shirt (black)	https://www.shirtinator.de/loadBasket/w2sVI72Xs18
Additional materials	...

Last Name	First Name	GitHub User Name	Email Address
Erben	Emanuel	emuguy1	emanuel.erben@fau.de
Nützel	Felix	Felix-012	felix.nuetzel@fau.de
Heimbs	Lennart	lheimbs	lennart.heimbs@fau.de
Böhm	Luca	QW3RAT	luca.boehm@fau.de
Malliaros	Nikolaos	nikomall34	niko.malliaros@gmail.com
Herzig	Tim Niklas	timherzig	tim.herzig@hotmail.com
Fogarty	Liam	lfogarty98	lfogarty9995@gmail.com
Oberson	Brianne	brianneoberson	brianne.oberson@gmail.com
Dargel	Olivia	oliviadargel	olivia.dargel@tu-berlin.de

#	Meeting Day	Product Owner	Software Developer	Release Manager	Scrum Master	Comment
1	2023-10-18	Nützel Felix, Emanuel Erben	Everyone else	N/A	Olivia Dargel	
2	2023-10-25	Nützel Felix, Emanuel Erben	Everyone else	Heimbs Lennart	Olivia Dargel	
-	2023-11-01	-	-	-	-	
3	2023-11-08	Nützel Felix, Emanuel Erben	Everyone else	Malliaros Nikolaos	Olivia Dargel	
4	2023-11-15	Nützel Felix, Emanuel Erben	Everyone else	Böhm Luca	Olivia Dargel	
5	2023-11-22	Nützel Felix, Emanuel Erben	Everyone else	Herzig Tim Niklas	Olivia Dargel	
6	2023-11-29	Nützel Felix, Emanuel Erben	Everyone else	Fogarty Liam	Olivia Dargel	
7	2023-12-06	Nützel Felix, Emanuel Erben	Everyone else	Oberson Brianne	Olivia Dargel	Mid-term due
8	2023-12-13	Nützel Felix, Emanuel Erben	Everyone else	Heimbs Lennart	Olivia Dargel	
9	2023-12-20	Nützel Felix, Emanuel Erben	Everyone else	Böhm Luca	Olivia Dargel	
10	2024-01-10	Nützel Felix, Emanuel Erben	Everyone else	Malliaros Nikolaos	Olivia Dargel	
11	2024-01-17	Nützel Felix, Emanuel Erben	Everyone else	Herzig Tim Niklas	Olivia Dargel	
12	2024-01-24	Nützel Felix, Emanuel Erben	Everyone else	Fogarty Liam	Olivia Dargel	
13	2024-01-31	Nützel Felix, Emanuel Erben	Everyone else	Oberson Brianne	Olivia Dargel	
14	2024-02-07	Nützel Felix, Emanuel Erben	Everyone else	Heimbs Lennart	Olivia Dargel	Demo day!
15	2024-02-14	Nützel Felix, Emanuel Erben	Everyone else	Böhm Luca	Olivia Dargel	Retrospective

Goals	Working Plugin that can be integrated in IntelliJ.
	We create a product that satisfies our industry partners
Meeting norms	We want to be punctual, if not, tell the group // maybe alternative: We start on time. If late, notify the others.
	Absence should be communicated before the meeting
	Focus and concentrate
Working norms	We finish our assigned tickets on time
	We upload our changes at 8 PM the day before our meetings
Coordination norms	We stick to our assigned roles
	Assign each task to a specific person.
Communication norms	No voice messages
	We communicate problems to each other
	Everyone checks the communication channel (Discord) regularly at least once a day.
	If someone is not reachable within 1 & 1/2 week, Prof. Riehle is informed and asked for further instructions
Consideration norms	We discuss disagreement openly
	We vote for final resolution
Cont. improvement norms	Teams progress will be tracked through weekly updates
Rewards	Everyone celebrates via a reaction in the zoom after each sprint
Sanctions	You have to complete unwanted tickets if you violate our norms
Signatures	
Scrum Master	Olivia Dargel
Product owner	Felix Nützel
Product owner	Emanuel Erben
Software developer	Lennart Heimbs
Software developer	Brianne Oberson
Software developer	Luca Böhm
Software developer	Liam Fogarty
Software developer	Nikolaos Malliaros
Software developer	Tim Niklas Herzig

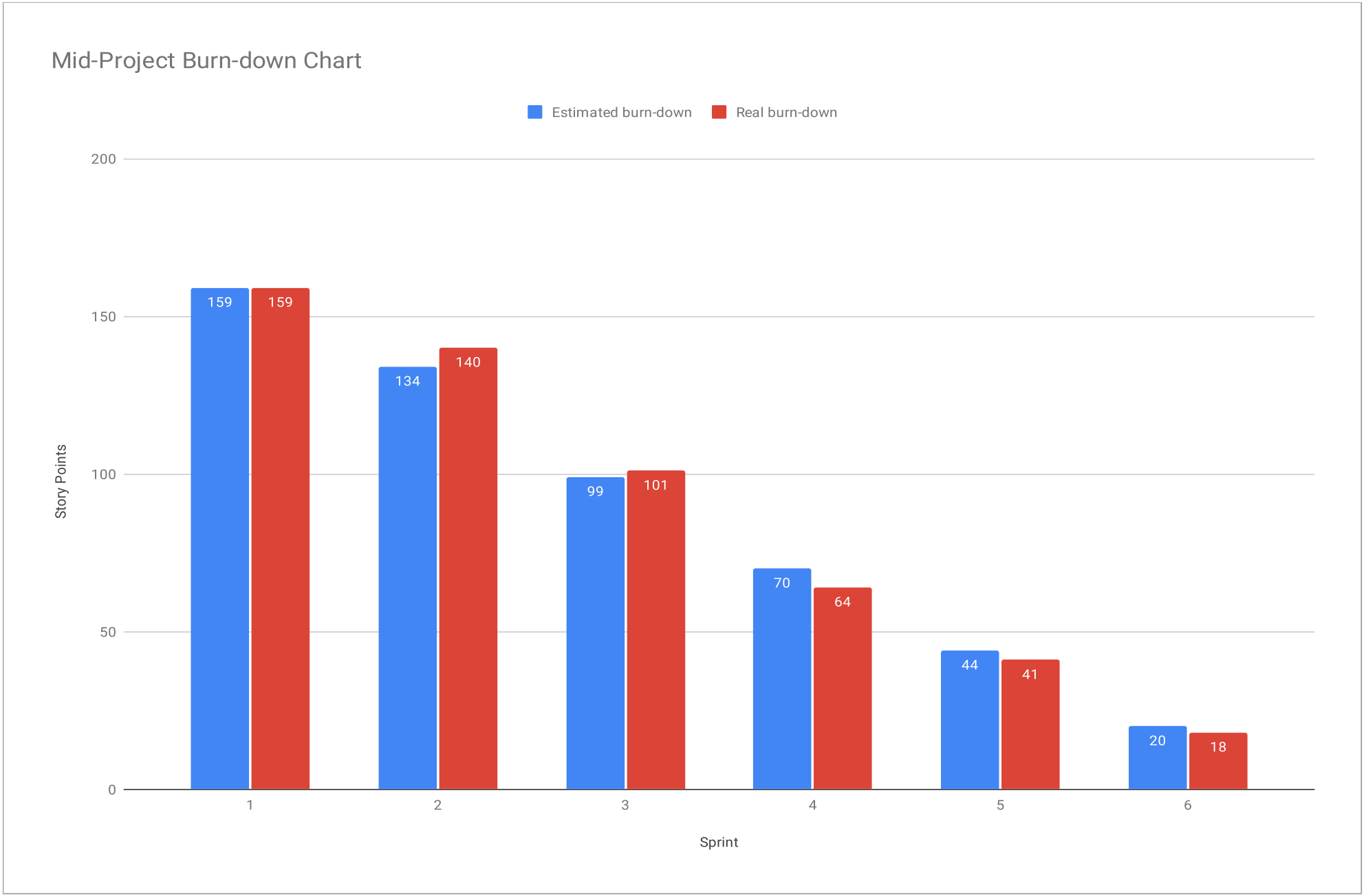
Product Vision	Project Mission
The reason of existence of the envisioned product (beyond this project):	The mission of this particular project (in the context of the product vision):
Software quality hinges on robust testing practices. While code coverage remains a prevalent metric, evaluating the true effectiveness of tests in ensuring expected behavior often gets overlooked. This is where Mutation Testing steps in—a method that generates code variations to evaluate the ability of tests to detect changes. PiTest, a leading tool in Mutation Testing, falls short due to its limited integration capabilities. It lacks the functionality to display test run results and configure test scope dynamically, creating a gap in assessing test effectiveness within the environment best known to the developer. Our product vision is to introduce an IntelliJ IDE plugin that not only presents PiTest results but also empowers users to seamlessly fine-tune test scopes, even down to specific classes. By integrating these features, we aim to bridge the existing gap, providing enhanced visibility and control within the familiar IntelliJ environment, thereby ensuring higher-quality test outcomes.	Our mission is to enhance software mutation testing within the IntelliJ IDE by implementing a specifically designed plugin that integrates with PiTest. The approach involves several key steps: Integration Development: We will develop an plugin that integrates with IntelliJ IDE, ensuring that PiTest's functionalities are easily accessible within the developer's primary workspace. Dynamic Test Configuration: A core feature of our plugin will be to enable dynamic configuration of test scopes. This will allow developers to selectively fine-tune their testing efforts, focusing on specific classes or modules. Result Visualization: The plugin will provide visualizations of Mutation Testing results. This will make it more comfortable for developers to interpret PiTest outputs. User-Centric Design: The interface and functionality of the plugin will be designed with a strong focus on user experience, ensuring that it is both powerful and easy to use. By following these steps, we aim to not only enhance PiTest’s functionality within IntelliJ IDE but also empower developers with more efficient, precise, and user-friendly software testing tools, ultimately leading to higher quality software development.

Term	Definition
Pitest/PIT	The Tool to run the Mutation Tests
Gradle	The build tool for Java/Android Applications which is used to run Pitest
Context Menu	The window that appears if you right click within the IDE
Color Bars	Bars that are shown at the beginnig of a line of code to indicate the result of a PiTest Run
Tool Window	The tool window contains 4 tabs. One for the Latest pitest result, the tree tab, a line chart and a bar chart.
Line Coverage	Defines how many lines of the source code are executed by the tests
Mutation Coverage	This is a measure of how many mutations in the code are detected and killed by the tests
Test Strength	This is an overall measure of how effective the test suite is in killing mutations
Coverage Report	Report generated by Pitest that includes Line Coverage, Mutation Coverage and Test Strength
Tree Tab	The tree tab contains the information from the coverage report sorted by packages for each class that is contained in the package

Sprint #	Sprint goal
1	None
2	None
3	None
4	Create first meaningful Features
5	Working with Test-Report Results
6	Create connection between our Plugin and Pitest Gradle Plugin
7	Create first working prototype with all core features
8	Start Publishing Process
9	Build more Visualizations
10	Add more customization
11	Refine Run Configurations
12	Techincal Refinments and Preperation Demo Day
13	Final Cleanup
14	Project Summary and Retrospective

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
Release						
Total			159	159		
Sprints						
1	**Research Basics**		25	159	19	159
2	**Software Architecture**		35	134	39	140
3	**Create Codebasics**		29	99	37	101
4	**Create first meaningfull Features**		26	70	23	64
5	**Working with Test-Report Results**		24	44	23	41
6	**Create connection between our Plugin and Pitest Gradle Plugin**		20	20	28	18
Features						
1	**Research Basics**					
		Research Mutation Testing	5		5	
		Research Plugin IntelliJ IDE	5		5	
		Initialize Readme.md and Wiki	5		2	
		Create team logo	5		5	
		Familiarize with Pitest	5		2	
2	**Software Architecture**					
		Create a Runtime Components Diagram	8		8	
		Create a Code Components Diagram	8		8	
		Create a Technology Stack Summary	5		3	
		Create a Textual Explanation of Diagrams and Choices	3		3	
		Initialize Software Bill of Material	3		2	
		Research best way to read PIT data	3		5	
		Create Code Skeleton	3		5	
		Create a Coding & Git Guideline	2		5	
3	**Create Codebasics**					
		Create a Build Guide for our Project	3		1	
		Obtain and Transform the Test Report	5		8	
		TestConfigurator that can interact with PiTest	8		13	
		Create a gradle connector that can interact with the project	5		5	
		Research possible configurations and parameters that can be forwarded to Pitest	3		2	
		Add Visualization for the User	5		8	
4	**Create first meaningfull Features**					
		Attend IntelliJ Webinar on how to get the Plugin into Marketplace on November 16th	2		2	
		Implement Context Menu for Class-Specific Run Execution	8		5	
		Color Bars on one side of the code indicating PiTest status	8		8	
		Research forward way of running Pitest	5		5	
		Develop and Document Basic Testing Framework for Plugins with Example	3		3	
5	**Working with Test-Report Results**					
		Research forward way of running Pitest	5		8	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		Create Build Process Video	3		3	
		Get the scope information to MutationMateRunConfiguration.kt	3		3	
		Implement Storing Old Pitest Runs	8		3	
		Improve GitHub CI Workflow	2		1	
		Load Pitest XML Files	3		5	
6	**Create connection between our Plugin and Pitest Gradle Plugin**					
		Initialize user, (technical) design and build/deploy documentation	3		3	
		Finish Gralde Plugin for Overwriting Settings	3		8	
		Start Pitest Plugin Run from IDE	2		3	
		Visualize Latest Pitest Report	3		5	
		Cleanup Wiki and separate into subpages	2		2	
		Connect Color Bars with data from PiTest	5		5	
		Create a Logo fitting for the Plugin	2		2	

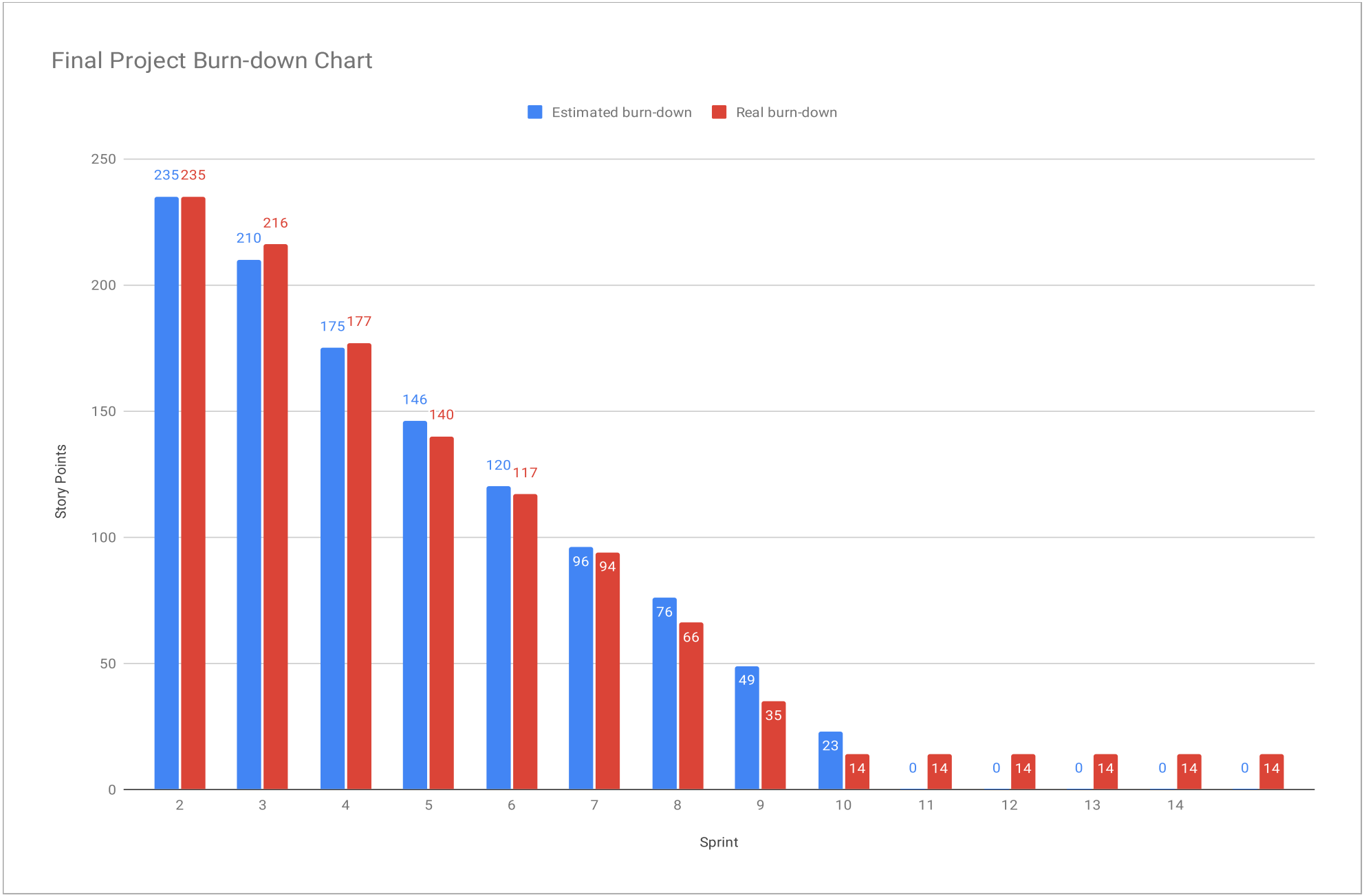

Mid-Project Burn-down Chart
Estimated burn-down
Real burn-down
200
150
100
50
0
Story Points
159
159
134
140
99
101
70
64
44
41
20
18
1
2
3
4
5
6
Sprint

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
Release						
Total			235	235		
Sprints						
				Estimated burn-down		Real burn-down
1	**Research Basics**		25	235	19	235
2	**Software Architecture**		35	210	39	216
3	**Create Codebasics**		29	175	37	177
4	**Create first meaningful Features**		26	146	23	140
5	**Working with Test-Report Results**		24	120	23	117
6	**Create connection between our Plugin and Pitest Gradle Plugin**		20	96	28	94
7	**Create first working prototype with all core features**		27	76	31	66
8	**Start Publishing Process**		26	49	21	35
9	**Build more Visualizations**		23	23	0	14
10	**Add more customization**		0	0	0	14
11	**Refine Run Configuration**		0	0	0	14
12	**Technical Refinments and Preperation Demo Day**		0	0	0	14
13	**Final Cleanup**		0	0	0	14
14	**Project Summary and Retrospective**		0	0	0	14
Features						
1	**Research Basics**					
		Research Mutation Testing	5		5	
		Research Plugin IntelliJ IDE	5		5	
		Initialize Readme.md and Wiki	5		2	
		Create team logo	5		5	
		Familiarize with Pitest	5		2	
2	**Software Architecture**					
		Create a Runtime Components Diagram	8		8	
		Create a Code Components Diagram	8		8	
		Create a Technology Stack Summary	5		3	
		Create a Textual Explanation of Diagrams and Choices	3		3	
		Initialize Software Bill of Material	3		2	
		Research best way to read PIT data	3		5	
		Create Code Skeleton	3		5	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		Create a Coding & Git Guideline	2		5	
3	**Create Codebasics**					
		Create a Build Guide for our Project	3		1	
		Obtain and Transform the Test Report	5		8	
		TestConfigurator that can interact with PiTest	8		13	
		Create a gradle connector that can interact with the project	5		5	
		Research possible configurations and parameters that can be forwarded to Pitest	3		2	
		Add Visualization for the User	5		8	
4	**Create first meaningful Features**					
		Attend IntelliJ Webinar on how to get the Plugin into Marketplace on November 16th	2		2	
		Implement Context Menu for Class-Specific Run Execution	8		5	
		Color Bars on one side of the code indicating PiTest status	8		8	
		Research forward way of running Pitest	5		5	
		Develop and Document Basic Testing Framework for Plugins with Example	3		3	
5	**Working with Test-Report Results**					
		Research forward way of running Pitest	5		8	
		Create Build Process Video	3		3	
		Get the scope information to MutationMateRunConfiguration.kt	3		3	
		Implement Storing Old Pitest Runs	8		3	
		Improve GitHub CI Workflow	2		1	
		Load Pitest XML Files	3		5	
6	**Create connection between our Plugin and Pitest Gradle Plugin**					
		Initialize user, (technical) design and build/deploy documentation	3		3	
		Finish Gralde Plugin for Overwriting Settings	3		8	
		Start Pitest Plugin Run from IDE	2		3	
		Visualize Latest Pitest Report	3		5	
		Cleanup Wiki and separate into subpages	2		2	
		Connect Color Bars with data from PiTest	5		5	
		Create a Logo fitting for the Plugin	2		2	
7	**Create first working prototype with all core features**					
		Improve Scope Info	3		5	
		HTML Parser of Pitest Report	5		5	
		Connect real test report data with the PiTest result window	3		3	
		Implement hover action	3		5	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		Create more tests	8		8	
		Error message if companion Gradle plugin for Mutation Mate is missingError message if companion Gradle plugin for Mutation Mate is missing	5		5	
8	**Start Publishing Process**					
		Publish Gradle Plugin	3		3	
		Start Publishing Process of IDE Plugin	3		2	
		Adjust colors according to Color Theme from IntelliJ	3		3	
		Show failed lines in the IntelliJ Problems Tab	5		5	
		Review runIDE IDE Version	2		2	
		Revisit Mutation Listener and possibility to replace Parser	5		5	
		Create more tests	5		1	
9	**Build more Visualizations**					
		Create a Tree Structure Tab for Test Results	5			
		Rework Color Bars and Hover Action	3			
		Improve not executable Error	3			
		Implement Version of Loading the Result from the Reports/Pitest	5			
		Adjust visualization for multiple classes	3			
		Research Maven running	3			
		Publish Gradle Plugin	1			
10	**Add more customization**					
		Check compatibility with android project with pitest configured				
		Option for enabeling/disabeling				
		Enable to have a complete folder as scope				
		Try to automatically create a run configuration if none exists yet				
		Improve PIT Run Button Design				
		Improve Saving of PiTest Reports				
		Rework Loading of Test Results to Happen on Project Opening				
11	**Refine Run Configuration**					
		Add scope to run configuration				
		Try to automatically create a run configuration if none exists yet				
		Create draft for Demo day presentation slides				

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		Get Scope Info to Maven				
		Release Jetbrains Plugin - Review from JetBrains				
		Add additional Chart for visualization				
12	**Technical Refinments and Preperation Demo Day**					
		Create one demo day slide				
		Create demo day video				
		Refactoring of bug intense code parts				
		Improve Test Coverage				
		Show acticve mutators				
		Add project summary to first tab				
13	**Final Cleanup**					
		create checklist for demo day presentation				
		update product glossary				
		finalize build documentation				
		finalize design documentation				
		finalize user documentation				
		optimise and clean up code				
14	**Project Summary and Retrospective**					
		Create project summary				
		Create Project retrospective				


Final Project Burn-down Chart
Estimated burn-down
Real burn-down
Story Points
Sprint
250
200
150
100
50
0
2
3
4
5
6
7
8
9
10
11
12
13
14
235
235
210
216
175
177
146
140
120
117
96
94
76
66
49
35
23
14
0
14
0
14
0
14
0
14
0
14

#	Feature Definition of Done	Sprint Release Definition of Done	Project Release Definition of Done
	All acceptance criteria are met.		
	Work products are uploaded to the Github repository.		
	A pull request is created for each related branch.		
	The work products in the pull requests are reviewed.		
	Github CI Workflow passes for the branches		
	The corresponding branches are merged and closed.		
	The bill of materials section of the planning documents is updated.		
	Tests are written for the added features if suitable		
		A working and significant enhancement from the previous sprint is designated as a release candidate.	
		Existing features and security protocols must remain operational.	
			The project can be successfully built and deployed
			All created tests are successful.
			Developer documentation is created.
			User documentation is created and updated
			The release has been approved by all team members
			The release has been approved by all team members
			All issues are closed
			All pull requests are closed

Type	Link / reference
General Documentation	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin/wiki
User Documentation	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin/wiki/User-Documentation
Technical Documentation	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin/wiki/Technical-Documentation
Design Documentation	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin/wiki/Design
Tech Stack	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin/wiki/Technology-Stack
Build/Deploy Documentation	https://github.com/amosproj/amos2023ws02-pitest-ide-plugin/wiki/Build--deploy-documentation

#	Context	Name	Version	License	Comment
software components					
	org.w3c.dom.Element	interface Element	Java Platform SE8	GNU GPL 2.0 with classpath exception	
	org.w3c.dom.Node	interface Node	Java Platform SE8	GNU GPL 2.0 with classpath exception	
	java.io.File	class File	Java Platform SE7	GNU GPL 2.0 with classpath exception	
	javax.xml.parsers.DocumentBuilderFactory	class DocumentBuilderFactory	Java Platform SE8	GNU GPL 2.0 with classpath exception	
	https://pitest.org/	Pitest	1.15.3	Apache License 2.0	
	com.netflix.nebula:nebula-test	Nebula test plugin	10.3.0	Apache License 2.0	
	https://jsoup.org/	Jsoup	1.17.1	MIT Licence	
libraries					
	https://plugins.jetbrains.com/docs/intellij/welcome.html	Intellij Plugin SDK	???	Apache License 2.0	
	commons-beanutils:commons-beanutils-core	Commons BeanUtils	1.8.3	Apache License 2.0	
	org.spockframework:spock-core	Spock Framework	2.2-groovy-3.0	Apache License 2.0	
tools					
	https://junit.org/junit5/	Junit 5	5.10.1	Eclipse Public License 2.0.	
	https://www.jetbrains.com/idea/	Intellij IDE Community	2023.2.4	Apache License 2.0	
developement process					
	https://groovy-lang.org/	Groovy	4.x	Apache License 2.0	
	https://kotlinlang.org/	Kotlin	1.9.x	Apache License 2.0	
build process	https://www.java.com/de/	Java JDK	>=17.0.1 and <21.0.0	Oracle Technology Network License Agreement	
	https://github.com/gradle	Gradle	8.6	Apache License 2.0	
publish process					

Last Name	First Name	Value					
Erben	**Emanuel**	5		**5.00**	**OK**		
Nützel	**Felix**	5					
Heimbs	**Lennart**						
Böhm	**Luca**						
Malliaros	**Nikolaos**			0	No size		
Herzig	**Tim Niklas**			1	Trivial size		
Fogarty	**Liam**			2	Small size		
Oberson	**Brianne**			3	Medium size		
Dargel	**Olivia**			5	Large size		
				8	Very large size		
				13	Too large (size)		