Final Handout

1 Statistics of efforts

Total hours spend on project: *364h 14m*

1.1 Hours per person, and the major contributions per person

Alexander Hagl	Anna Weber	Jasmin Hübner	Nick Hübner
139h 35m	74h 45m	99h 54m	50h
Documentation	Documentation	Documentation	Documentation
Blog	Blog	Blog	Design
Setup and CI/CD	Scrum-master	Mock-ups	Mock-ups
Multiple services in back-end	Multiple services in front-end and back-end	UserService back-end	Services in front-end
Some views in front-end	Authentication	Statistics View	Multiple views in front-end
Refactoring	Clean code	Metrics analysis	Refactoring
Architecture	Mail-Service	Leader-board service front-end	
Testing		Testing	
Benchmarking			

1.2 Hours per workflow

Workflow	Total
Business Modeling	9h 20m
Project management	28h 55m
Requirements	22h 58m
Analysis & Design	33h 14m
Implementation	157h 10m
Test	30h 55m
Deployment	29h 15m
Quality Assurance	44h 55m
Learning	7h 32m

1.3 Hours per phase

Phase	Total
Inception	14h 45m
Elaboration	59h 17m
Learning	7h 32m
Construction	193h 38m
Back-end	101h 11m
Front-end	149h 7m
Transition	61h 37m
Blog	10h 10m
Presentation	11h 45m

2 Demo-Highlights

2.1 Landing Page (with leader-boards) when logged in:
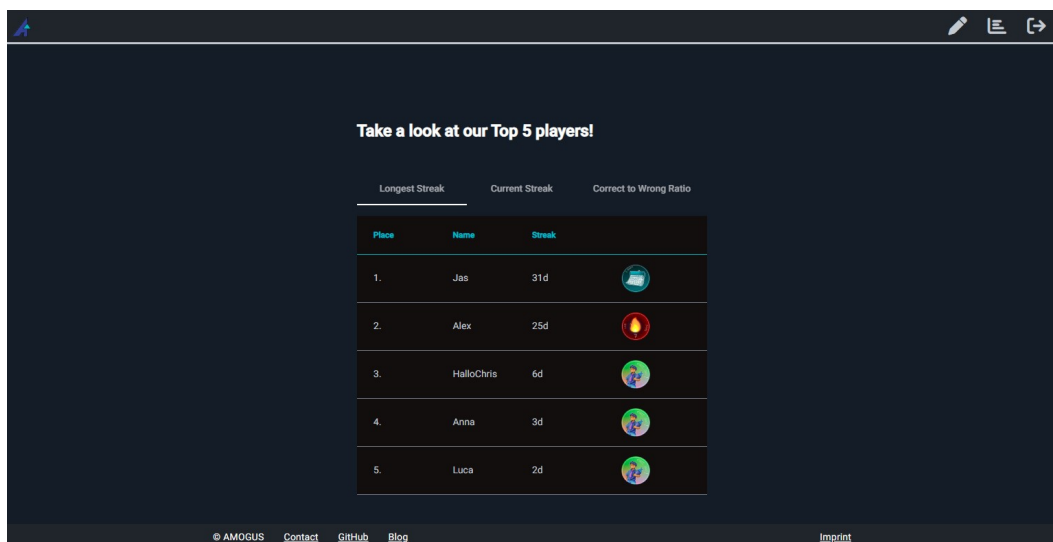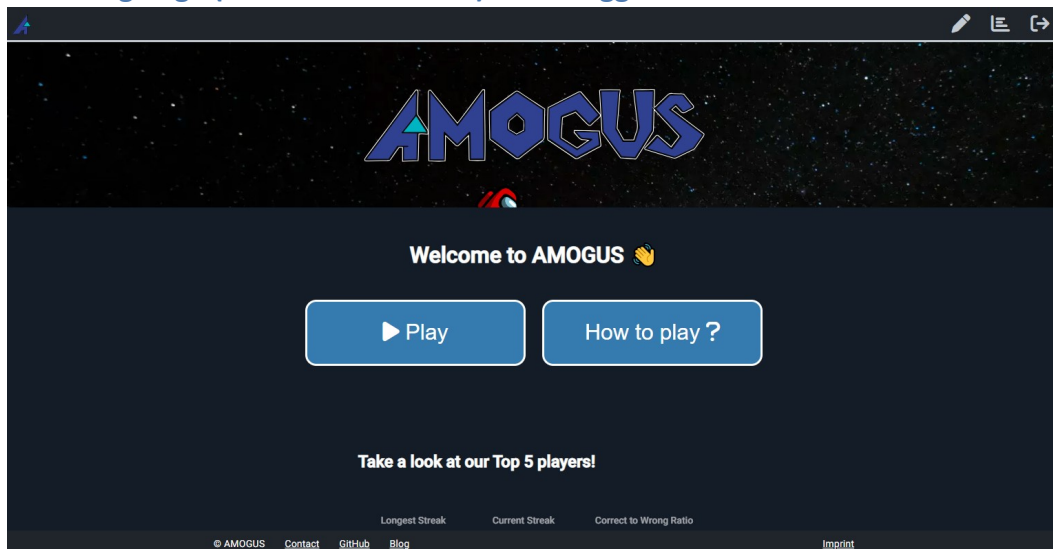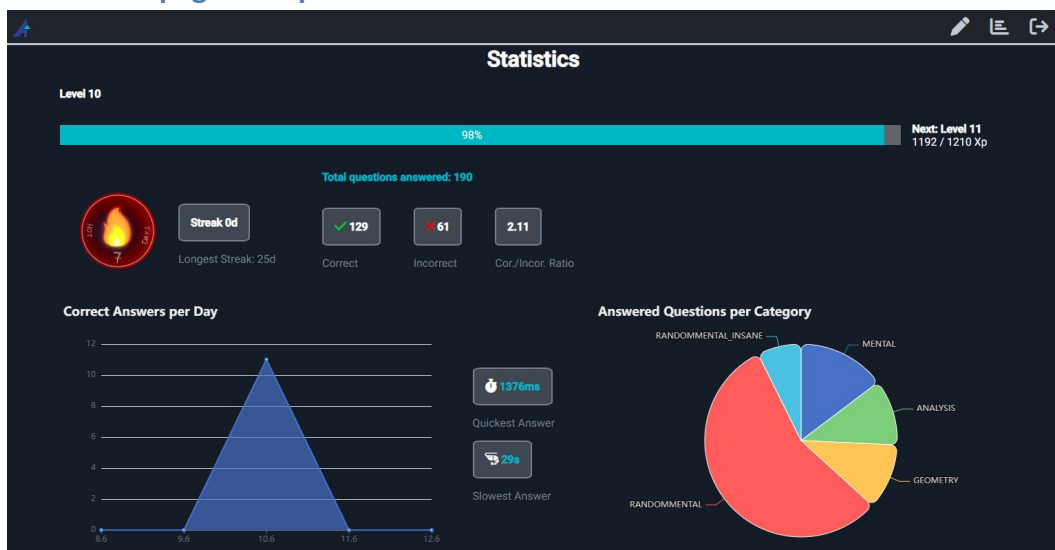

2.2 Game selection and game play:
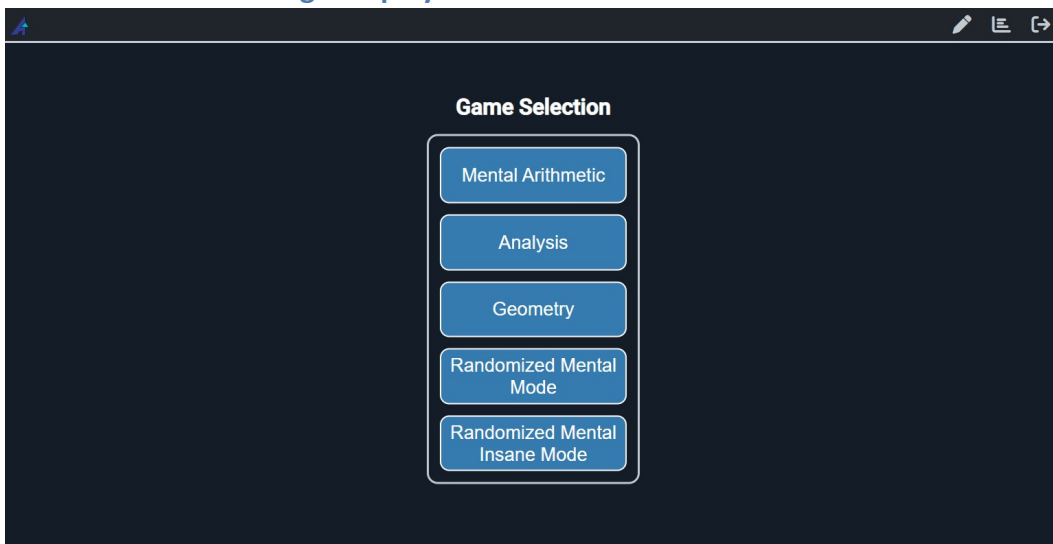
a) Game modes with preset questions (e.g. Analysis):

Find the limit of the function $f(x) = (x^2 - 1) / (x - 1)$ as x approaches 1.

- ○ 0
- ○ 3
- ○ 2
- ○ 1

Submit

b) Game modes with random generated questions (e.g. Random Mental Mode):

$7 + 8$

Submit
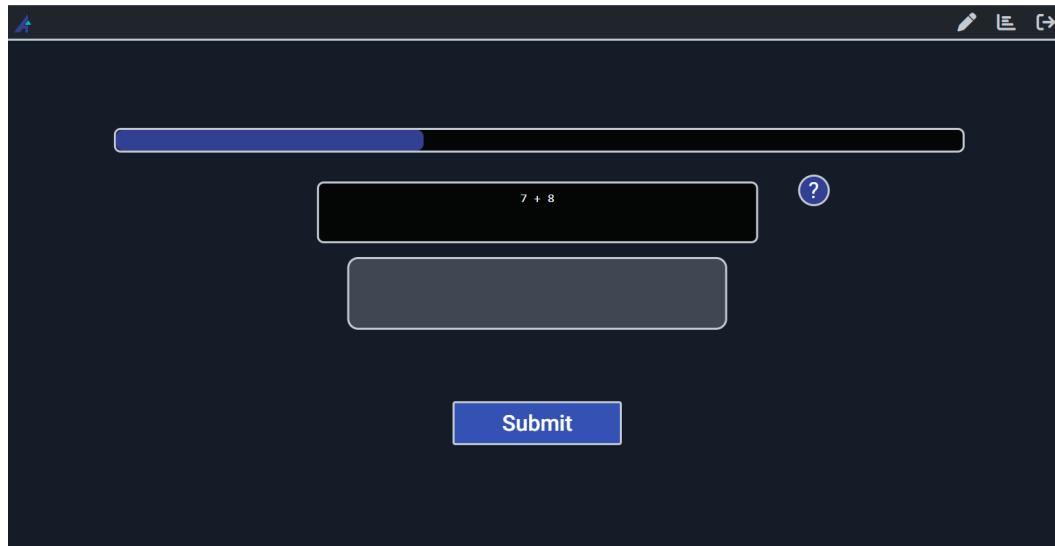
2.3 Statistics page with personal statistics:

Statistics

Level 10

98%   Next: Level 11
1192 / 1210 Xp

Total questions answered: 190

Streak 0d
Longest Streak: 25d

✓ 129   ✗ 61   2.11
Correct   Incorrect   Cor./Incor. Ratio

Correct Answers per Day

⏱ 1376ms
Quickest Answer

🐌 29s
Slowest Answer

Answered Questions per Category

RANDOMMENTAL_INSANE
MENTAL
ANALYSIS
GEOMETRY
RANDOMMENTAL

2.4 Teacher view to view and edit the preset questions-set:

3 Project Highlights

3.1 Architecture

Layered- and Service-Based Architecture

- AMOGUS follows a service based architecture to enforce modularization and increase cohesion, hence making modifications more centralized and independent.
- Coupling is reduced by following the "ABBA-Scheme" and only allowing communication via well defined interfaces.
- Back-end and front-end are completely separated (only communicating through a REST-API) to further decouple the application.
- Back-end and front-end are deployed in different docker containers therefore making deployment easier.

3.2 Tech-Stack

Back-end	Database	Front-end	Project management	Blog	Deployment	Libraries
C#, ASP.Net Web API, Visual Studio 2022	Dev.: SQLite Prod.: MariaDB	Angular, Visual Studio Code	Jira, Discord, GitHub	GitHub Discussions	Docker, Ubuntu, GitHub Actions	xUnit, Moq, AnguriMath

3.3 Database design
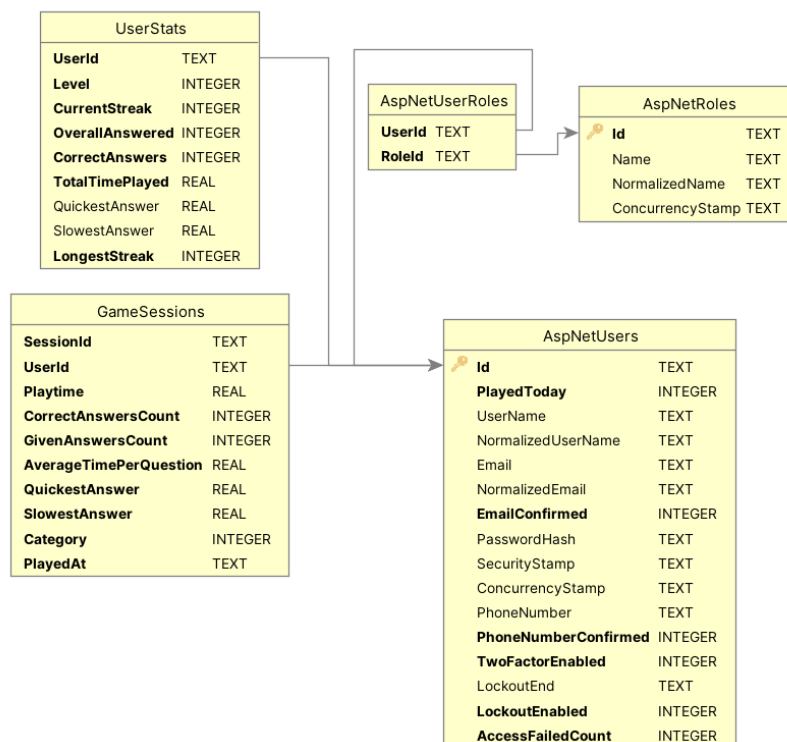
3.4 Testing

- Unit testing using the xUnit and the Moq framework. Using the GitHub Actions all unit-tests are run with each push to the repository (over 100 test-cases were written).

- Three bugs were found while testing (all are fixed):

    - UserService – DeleteUserAsync() returned an exception when deletion succeeded.

    - MentalExerciseFactory – CalcAnswer() threw an exception that wasn't caught, when question contained 'x' but wasn't solvable.

    - MentalExerciseFactory – Median() was implemented wrong.

- Target test coverage was 60% → Achieved coverage can be found in 3.5 Measurements

- Beta testing by letting family and friends try out the website.

3.5 Measurements

a) Source Code Complexity and Class Coupling of the back-end, analyzed by Visual Studio 2022:

Visual Studio aggregates the metrics of the functions of a class and provides this as a result on a class level. This leads to higher values in these metrics. To calculate it more precise the results from Visual Studio were taken on a class level and for the assembly level the median of the class level values were taken. Following table shows the median of the assembly level values:

	Code Complexity	Class Coupling
Project AMOGUS (median)	5	10

b) Web Application Metrics of the front-end, analyzed with Lighthouse:

| Front-end	Before optimizing images	After optimizing images
Landing Page	77 Leistung / 86 Barrierefreiheit / 100 Best Practices / 73 SEO / PWA	90 Leistung / 86 Barrierefreiheit / 100 Best Practices / 73 SEO / PWA
Statistics Page	75 Leistung / 87 Barrierefreiheit / 100 Best Practices / 82 SEO / PWA	92 Leistung / 87 Barrierefreiheit / 100 Best Practices / 82 SEO / PWA
How-To-Play Page	91 Leistung / 83 Barrierefreiheit / 100 Best Practices / 82 SEO / PWA	93 Leistung / 83 Barrierefreiheit / 100 Best Practices / 82 SEO / PWA

c) Code- and Branch-Coverage from unit testing:

Information		Line coverage		Branch coverage	
Parser:	Cobertura	**91%**	Covered lines: 431	**83%**	Covered branches: 117
Assemblies:	3		Uncovered lines: 38		Total branches: 140
Classes:	12		Coverable lines: 469		Branch coverage: 83.5%
Files:	12		Total lines: 971		
Coverage date: 14.05.2023 - 16:15:23			Line coverage: 91.8%		

Coverage history
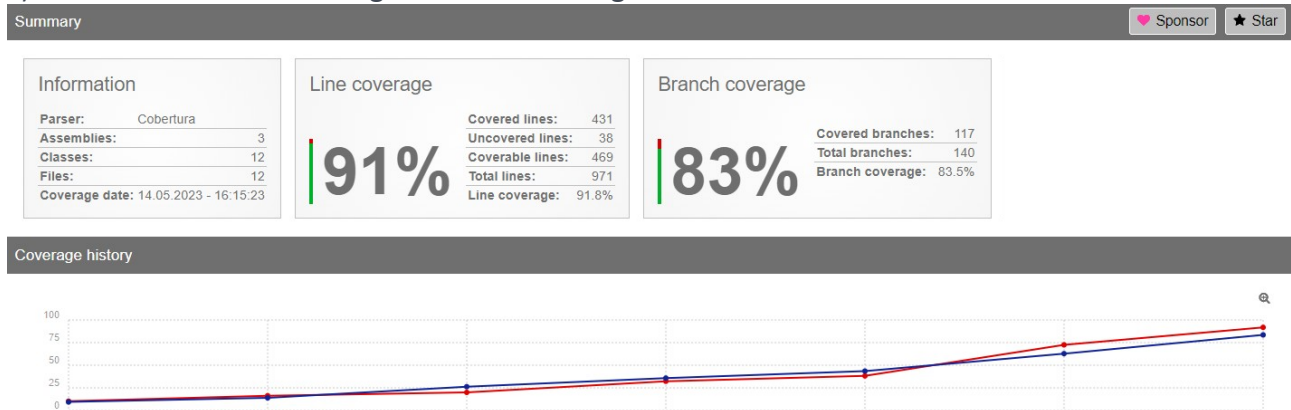
3.6 CI/CD

a) Continuous Integration on each pull-request by running two GitHub Action workflows:
- Build projects
- Run unit-tests

If any of those fail, the pull-request is rejected.

b) Continuous Deployment on each push to the main-branch by conditionally running two GitHub workflows:
- When **src/api** is modified: Build **API-Docker-Image**, upload image to DockerHub and send webhook to server.
- When **src/ui** is modified: Build **UI-Docker-Image**, upload image to DockerHub and send webhook to server.