HTW Berlin
Department of Computer Science

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Name: _____ Matriculation number: _____

Your grade in the following exam does not reflect your talent as a fullstack developer. Many skills that are essential for every developer cannot be assessed in a written exam, e.g. self-organization, communication of knowledge, empathy, helping out your co-workers.

The primary purpose of this exam is to generate an objective metric for your grade which is a formal requirement of the university.

Besides that, the exam may work as a motivation for you to learn: When you read this sentence you are probably preparing yourself for the exam, so you learn something along the way.

Good luck!
Robert Schäfer

Question:	1	2	3	4	5	6	7	8	Total
Points:	10	11	8	8	9	11	4	10	71
Score:									

1. Project Management

[1] (a) Complete the following sentence: The motto of our course is “Le___n and S_____e”

[1] (b) What is the bus factor? Write a definition.

.....

.....

.....

[1] (c) What is better: A high or a low bus factor?

.....

[2] (d) In the lecture we covered five benefits of software testing. Please fill in at least two of them in figure 1 and explain them by example: What could go wrong without software testing with respect to that benefit? Why does something work better with software testing?

.....

.....

.....

.....

.....

.....

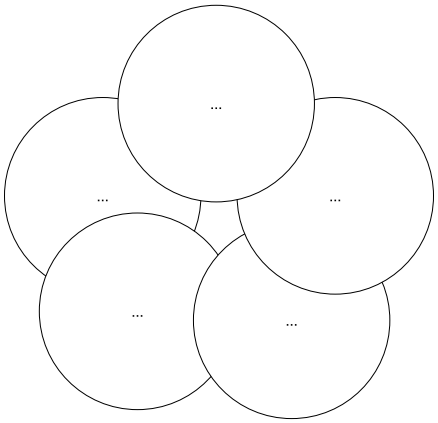


Figure 1: Five benefits of software testing

[3] (e)

There will be at least three more questions here

[2] (f)

There will also be a question about continuous integration

2. Git

[1] (a) What data is hashed in order to generate a git commit id? Enumerate at least three attributes.

.....

.....

[2] (b) Is `git reset --hard` reversible? If yes, how can you reverse it? If no, please explain why.

.....

.....

.....

.....

[2] (c) Is `git clean --force` reversible? If yes, how can you reverse it? If no, please explain why.

.....

.....

.....

.....

(d) Consider the following scenario: You forgot to create a feature branch and committed on branch `master` by accident.

Now you want to create a pull request. In order to do that you create a new branch with:

```
git checkout -b feature-branch
```

You can see your current git history in figure 2.

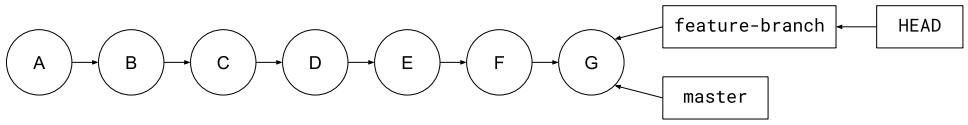


Figure 2: After `git checkout -b feature-branch`

[1] i. What are the commands that produce a git history as shown in figure 3?

.....

.....

.....

.....

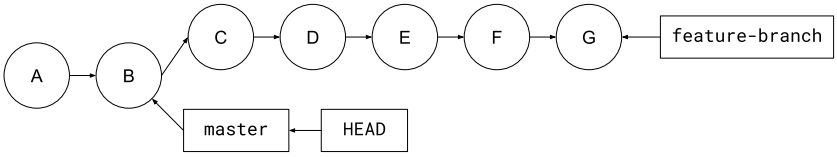


Figure 3: Remove commits from `master`

1 ii. Consider your git history looks as shown in figure 3. On wich commit do you need to rebase in order to produce a git rebase log as shown in figure 5 and finally a git history as shown in figure 4?

```
git checkout feature-branch
git rebase --interactive ....
```

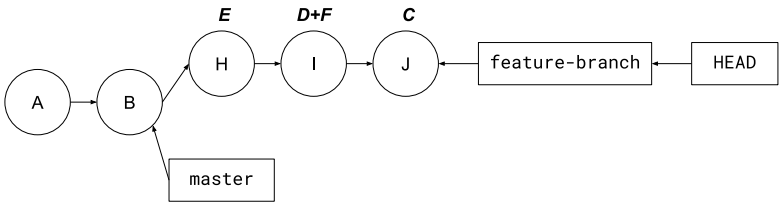


Figure 4: Commits contain changes of the commits on top of the respective nodes

2 iii. How do you need to change the rebase log in figure 5 to produce a git history as shown in figure 4?

```
pick C      ........................
pick D      ........................
pick E      ........................
pick F      ........................
pick G      ........................
```

Figure 5: Rebase log of your editor

2 (e) TODO

In the final exam, I will create a more elaborate scenario and assign more points to it

3. Functional Programming

[1] (a) What are higher order functions?

.....

[1] (b) Give an example of a higher order function in JavaScript:

(c) The following code examples have unnecessary temporary variables. How can you refactor the code examples and eliminate these temporary variables?

```
function findNextId (){
  let lastId = 0;
  for (i = 0; i < data.todos.length; i++) {
      if (data.todos[i].id > lastId) {
          lastId = data.todos[i].id;
      }
  }
  lastId += 1;
  return lastId;
}
```

```
function filterTodos(mok, userAuth) {
  var retArray = []

  for (var i = 0; i < mok.length; i++) {
      if (mok[i].userAuth == userAuth) {
          retArray.push(mok[i])
      }
  }
  return retArray
}
```

Add one more code example here

3 i. You will get a point per code example if you can name the right method which you could apply.

3 ii. You will get a point per code example if you can write down the correct (or almost correct) refactored source code.

4. GraphQL and Apollo-Server

[1] (a) What are the problems that GraphQL solves in comparison with REST? Explain how exactly GraphQL solves these problems.

.....

.....

.....

.....

[2] (b) Look at the implementation in figure 7. When we send the graphql query in figure 6 to the server, what is the response of the server?

```
query {
  hello(name: "Again")
}
```

Figure 6: Graphql query

(c) What gets written to the terminal of the backend? Write down the output of the `console.log` statements. You will get points for the following:

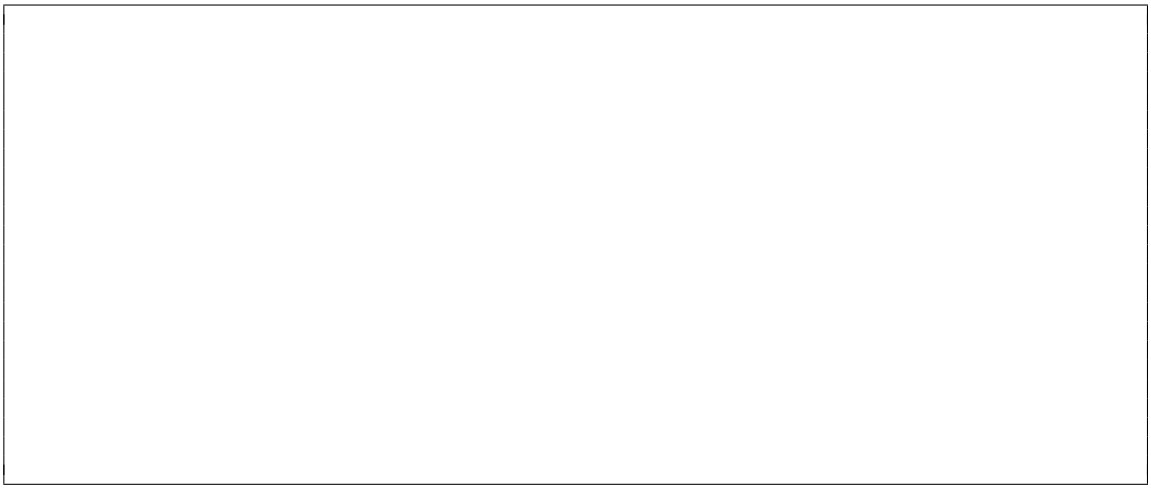
[1] i. Correct selection of traversed calls

[1] ii. Correct number of calls

[1] iii. Correct order of calls

[1] iv. Correct value of `parent`

[1] v. Correct value of `args`

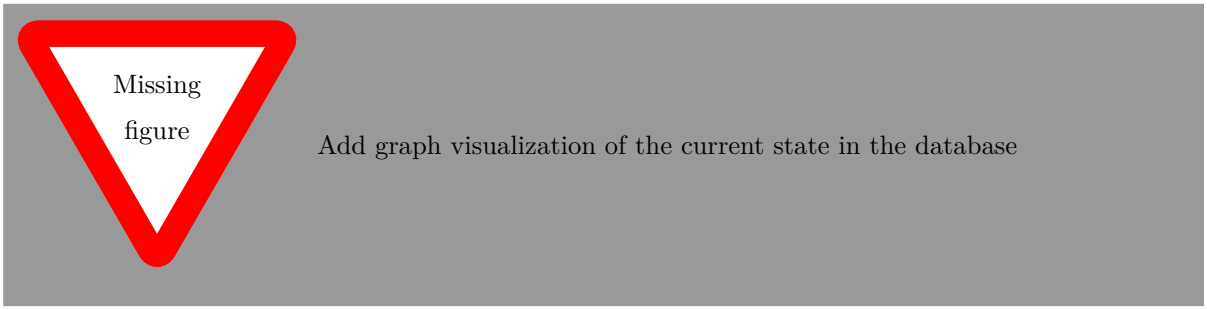


```
const { applyMiddleware } = require('graphql-middleware')
const { ApolloServer, gql } = require('apollo-server')
const { makeExecutableSchema } = require('graphql-tools')

const typeDefs = `
type Query {
  hello(name: String): String
  bye(name: String): String
}
`

const logInput = async (resolve, root, args, context, info) => {
  console.log(`1. logInput: ${JSON.stringify(args)}`)
  const result = await resolve(root, args, context, info)
  console.log(`5. logInput`)
  return result
}

const logResult = async (resolve, root, args, context, info) => {
  console.log(`2. logResult`)
  const result = await resolve(root, args, context, info)
  console.log(`4. logResult: ${JSON.stringify(result)}`)
  return result
}

const middlewares = [logInput, logResult]

const resolvers = {
  Query: {
    hello: (root, args, context, info) => {
      console.log(`3. resolver: hello`)
      return `Hello ${args.name ? args.name : 'world'}!`
    },
    bye: (root, args, context, info) => {
      console.log(`3. resolver: bye`)
      return `Bye ${args.name ? args.name : 'world'}!`
    },
  },
}

let schema = makeExecutableSchema({ typeDefs, resolvers })
schema = applyMiddleware(schema, ...middlewares)
const server = new ApolloServer({ schema });
server.listen().then(({ url }) => {
  console.log (`  Server ready at ${url}`);
});
```

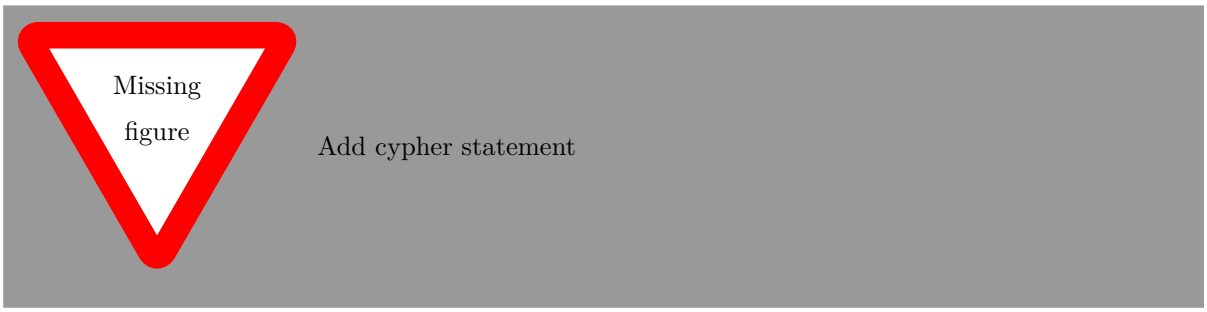
Figure 7: Apollo server implementation

5. Neo4J

Here is the visualization of a graph.



When you send the following cypher statement to the database:

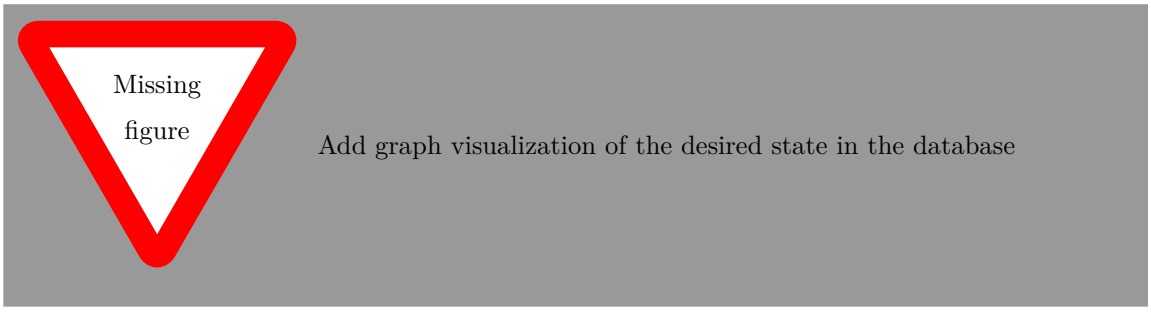


3 (a) How does the result set look like?

3 (b) How many records are in the result set?

.....

3 (c) Here is a graph visualization of a desired state in the database.



Write down a single cypher statement that produces the desired result.

6. VueJS and NuxtJS

[2] (a) What are the pros and cons of a single page application and client-side-rendering in comparison with a traditional multi-page and server-side-rendered application?

.....

.....

.....

.....

[1] (b) What is isomorphic code?

.....

.....

[2] (c) NuxtJS provides server-side-rendering for VueJS. What is the motivation to do both server-side-rendering and client-side-rendering?

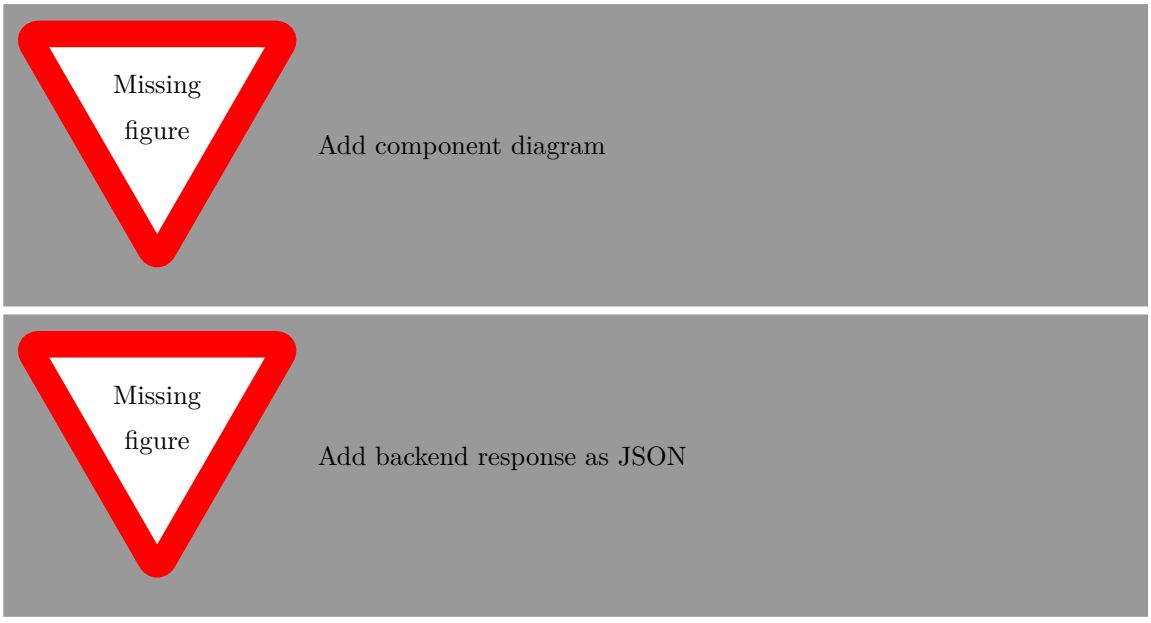
.....

.....

.....

.....

(d) Have a look at this diagram which shows the different components and its responsibilities.



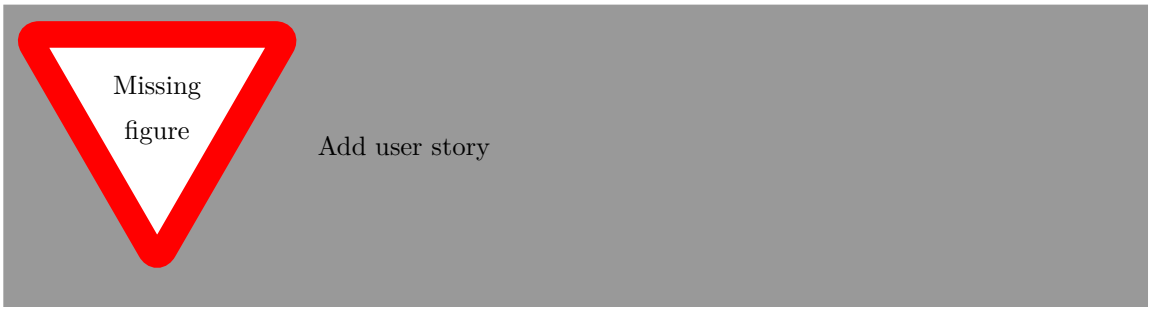
[3] i. What is the minimal data that needs to get passed as property down to the subcomponents respectively? Write your answer as an annotation in the diagram.

[3] ii. What are the events that should get emitted from the subcomponents to the parent components respectively? Write your answer as an annotation in the diagram.

7. Fullstack testing and Requirements Engineering

Add description of a web application in detail

2 (a) What are the issues with the following user story?



2 (b) Write down a cucumber scenario which could be a valid acceptance criteria for the user story.

8. Code Review

10

(a) Give a code review of the following pull request. Find and annotate issues in the code. You will get a point only if you can explain what the problem is. You can get another point if you can give a suggestion how to fix the problem. There are about 20 issues in the code example. It is enough if you can find 10 issues and explain the problem. Alternatively you can also find 5 issues, explain the problem **and** write down the source code as a suggestion to fix it. Anything in between is also possible.

- Mixing async/await with then/catch
- Not sanitizing strings and having a cypher injection vulnerability
- C-style for loop
- Writing 'should' in test case descriptions
- Not closing sessions
- Multiline string without backticks
- Using index literals or keys instead of array/object destructuring
- Merge objects with ... spread operator
- Secrets in source code
- Not using jest's async expectations
- Nested if-clauses instead of guard clauses
- Expose user listing in login resolver
- Missing ID in type definitions
- Not using hard-coded values for expectations