Name: _____ Matrikelnummer: _____

Deine Note in der Endklausur sagt nichts über dein Talent als Fullstack-EntwicklerIn aus. Viele wesentliche Fähigkeiten können nicht in einer schriftlichen Prüfung bewertet werden, z.B. selbstständiges Lernen, Wissensvermittlung, Einfühlungsvermögen und Hilfsbereitschaft innerhalb des Teams.

Der primäre Zweck der Endklausur ist es, eine formelle Anforderung der Universität zu erfüllen, nämlich zu einer quantifizierbare Metrik pro Teilnehmer zu finden.

Abgesehen davon dient die Klausur als Motivation: Wenn du diesen Satz liest, dann bereitest du dich wahrscheinlich auf die Prüfung vor. In diesem Fall hat die Endklausur und die Probeklausur ihren Zweck erfüllt.

Viel Erfolg!
Robert Schäfer

Question:	1	2	3	4	5	6	7	8	9	10	Total
Points:	16	9	10	4	16	11	18	5	4	10	103
Score:											

1. Project Management

Mentoring

Knowledge transfer

Code Reviews

Testing

Continous Integration

[1] (a) Vervollständige den folgenden Satz: Das Motto unseres Kurses ist: “Le___n and S____e”

[1] (b) Was ist der Busfaktor? Schreibe eine Definition auf.

.....

.....

[1] (c) Was ist besser? Ein möglichst hoher Busfaktor oder ein niedriger Busfaktor?

A. Ein hoher Busfaktor ist besser.

B. Ein niedriger Busfaktor ist besser.

[3] (d) Wie kann der Wissenstransfer im Team gewährleistet werden? Nenne mindestens drei unterscheidbare Methodiken, die wir im Kurs kennen gelernt haben:

.....

.....

.....

[4] (e) Wobei helfen automatisierte Software-Tests?

Software-Tests helfen dabei:

- ◯ Regressionen zu vermeiden
- ◯ das Verhaltens der Anwendung zu dokumentieren
- ◯ Abhängigkeiten im Quellcode zu erkennen und zu vermeiden
- ◯ bei der Software-Architektur, etwa bei beim Design von Schnittstellen
- ◯ bei der Wartung des Quellcodes, z.B. beim Refactoring
- ◯ die Korrektheit des Quellcodes zu beweisen
- ◯ sicherzustellen, dass die Anforderungen des Kunden bzw. Endbenutzers erfüllt sind
- ◯ Aussagen über das Laufzeitverhalten zu treffen, z.B. Komplexitätsklassen von Algorithmen

(f) Wir haben in unserem Kurs einen Software-Test als “positiv” definiert, wenn er fehlschlägt. (Also ähnlich wie ein “positiver” medizinischer Test, welcher aussagt, dass eine Testperson mit einer Krankheit infiziert sei.)

Fehlerhafte Software-Tests können zu folgenden Problemen führen:

A. Das Team verliert an Disziplin und das Vertrauen in Software-Testing. Es beginnt Änderungen am Quellcode zu akzeptieren, obwohl der Build-Server fehlgeschlagen ist.

B. Es kommt zu Verzögerungen beim Ausrollen der Anwendung (Deployment).

C. Die Endbenutzer können betroffene Funktionen der Anwendung nicht mehr nutzen.

D. Es können Sicherheitslücken entstehen.

[2] i. Welche dieser Probleme entstehen durch "falsch-positive" Software-Tests? Auswahl: _____

[2] ii. Welche dieser Probleme entstehen durch "falsch-negative" Software-Tests? Auswahl: _____

[2] (g) Mit welcher Technik kann man "falsch-negative" Tests vorbeugen? Anders gefragt: Wie kann man sicherstellen, dass ein Software-Test überhaupt einen Fehler aufzeigen würde?

.....

.....

2. Git

git rebase

git clean

git reflog

git reset

git blame

git remote

git checkout <REFERENCE> – path/to/file

Difference between working tree and index

How to write better commit messages

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

.....

.....

[2] (b) Is `git reset <ID> --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)

We will do the following exercise the other way around: Instead of you writing down the git commands that produce a given history, I will give you some commands and your task is to draw the resulting git history.

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 1.

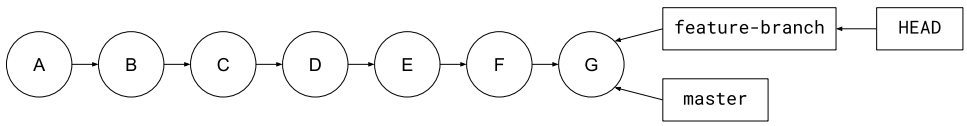

Abbildung 1: After `git checkout -b feature-branch`

1 i. What are the commands that produce a git history as shown in figure 2?

.....

.....

.....

.....

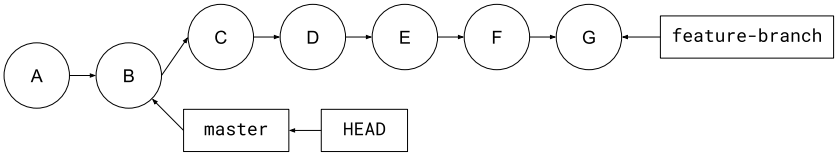

Abbildung 2: Remove commits from `master`

1 ii. Consider your git history looks as shown in figure 2. On wich commit do you need to rebase in order to produce a git rebase log as shown in figure 4 and finally a git history as shown in figure 3?

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

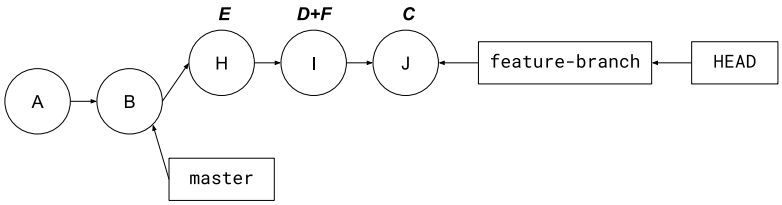

Abbildung 3: Commits contain changes of the commits on top of the respective nodes

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

Abbildung 4: Rebase log of your editor

2 iii. How do you need to change the rebase log in figure 4 to produce a git history as shown in figure 3?

3. Funktionale Programmierung

2 (a) Was sind "higher order functions"?

.....

2 (b) Im folgenden Code-Beispiel wird eine Funktion namens `higherOrderFunction` deklariert, die Argumente der Funktion heißen `some, arguments, you, can` und `choose`. Bitte schreibe eine Implementierung, sodass die Funktion *ganz offensichtlich* zu einer "higher-order function" wird.

```
function higherOrderFunction(some, arguments, you, can, choose) {
  // Write down any implementation, which obviously turns this function into a
  // higher order function.
  // ...




 // return ?
 // If you want, return something
}
```

(c)

Lerne folgende Methoden auf `Array.prototype`: `forEach`, `map`, `find`, `filter`, `reduce`

Die folgenden Code-Beispiele enthalten unnötige temporäre Zustände. Zustände sind immer schlecht, weil sie mögliche Fehlerquellen sind. Wie können die folgenden Code-Beispiele umgeschrieben (refactored) werden, um die unnötigen Zustände zu eliminieren? Das Verhalten des Codes darf sich nicht verändern.

```
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
}
```

3 i. Es gibt für jedes Code-Beispiel einen Punkt, wenn die richtige Funktion von `Array.prototype` ausgewählt wird.

3 ii. Für jedes korrekte Refactoring eines Code-Beispiels gibt es einen weiteren Punkt. Geringfügige Syntax-Fehler führen zu keinem Punktabzug.

4. Authorization and Authentication

2 (a) Was bedeuten die Begriffe “Authorization” und “Authentication”? Erkläre, welche Aufgaben mit

den Begriffen gemeint sind und grenze sie voneinander ab.

. . . . . .

. . . . . .

. . . . . .

[1] (b) Richtig oder Falsch: Die Nutzdaten (der Payload) eines JWT (JSON-Web-Token) werden verschlüsselt übertragen.

A. Richtig

B. Falsch

[1] (c) Richtig oder Falsch: Wenn der Schlüssel zur Überprüfung der Signatur der JSON-Web-Token erneuert wird, werden alle bis dahin ausgelieferten Token invalidiert.

A. Richtig

B. Falsch

5. GraphQL and Apollo-Server

[7] (a) Welche Probleme löst GraphQL in Bezug auf REST?

◯ HTTP-Caching Mechanismen werden besser ausgenutzt .

◯ Auf dem Client kann die Benutzeroberfläche automatisch aktualisiert werden.

◯ Die Anzahl der Anfragen wird minimiert.

◯ Die Menge der ausgetauschen Daten pro Anfrage (der Traffic) wird minimiert.

◯ Die Anzahl der Datenbank-Abfragen wird minimiert.

◯ Daten können in einer Graph-Datenbank gespeichert werden.

◯ Der Server wird robuster gegenüber Denial-of-Service Angriffen.

(b) Die Methodensignatur eines resolvers enthält vier Argumente, nämlich `parent`, `args`, `context` und `resolveInfo`.

[1] i. Sollte das Argument `context` in allen resolvern gleich sein?

A. Ja

B. Nein

[1] ii. Wozu wird das Argument `resolveInfo` genutzt?

. . . . . .

. . . . . .

[2] (c) Schau auf die Implementierung eines Apollo-Servers in Abbildung 6. Wenn wir die GraphQL Anfrage in Abbildung 5 an den Server schicken, mit welcher Antwort reagiert der Server?

```
query {
  allStudents(limit: 2) {
    id
    fullname(reverse: true)
  }
}
```

Abbildung 5: Graphql query

(d) Was schreibt der Server als Ausgabe aufs Terminal? Schreibe die `console.log` Ausgaben auf. Folgende korrekte Angaben werden bewertet:

[1] i. Korrekte Auswahl, welche Log-Ausgaben überhaupt erreicht werden

[1] ii. Korrekte Anzahl aller Ausgaben

[1] iii. Korrekte Reihenfolge der Ausgaben

[1] iv. Korrekter Wert des Arguments `parent`

[1] v. Korrekter Wert des Arguments `args`

6. Neo4J

(a) Wie wird das relationale Datenmodell in ein graph-basiertes Datenmodell übertragen? Nimm als Beispiel eine Neo4J-Datenbank.

[1] i. Der Name einer Tabelle wird zu: _____

1 ii. Die Spalten in einer Tabelle werden zu: _____

1 iii. Eine Zeile in einer Tabelle wird zu: _____

1 iv. Fremdschlüssel zu anderen Tabellen werden zu: _____

(b) Betrachte die `cypher` query:

```
MATCH (node)-[*]->(yetAnotherNode) RETURN *
```

i. Was bedeutet der Stern in den eckigen Klammern im "pattern" dieser query?

.....

.....

ii. Ist so eine Anfrage auch in einer relationalen Datenbank mit SQL möglich?

A. Ja

B. Nein

2 (c) Können in einer Neo4J Datenbank Fremdschlüssel auf fehlende Einträge zeigen? Solche Zeiger ins Leere heißen auch "dangling pointer". Begründe deine Antwort:

.....

.....

1 (d) Wieviele "Labels" darf ein Knoten in einer Neo4J Datenbank haben? Antwort: _____

1 (e) Wieviele "Typen" darf eine Beziehung in einer Neo4J Datenbank haben? Antwort: _____

(f) Consider the cypher statements:

```
CREATE(romeo:Person {name: 'Romeo'})
CREATE(juliet:Person {name: 'Juliet'})
```

Figure 7 shows how the graph looks initially.

When you run the following statement:

```
MERGE(:Person {name: 'Romeo'})-[:LOVES]->(:Person {name: 'Juliet'})
```

The graph looks like shown in figure 8.

1 i. How can you fix the cypher statement so that we don't get duplicate nodes?

2 ii. What has happened, why have duplicate records been created at all? Can you explain this?

.....

.....

.....

.....

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

const typeDefs = `
type Student {
  id: ID
  firstname: String
  lastname: String
  fullname(reverse: Boolean): String
}

type Query {
  student(id: ID): Student
  allStudents(limit: Int): [Student]
}
`
const students = [
  { id: 1, firstname: 'Alice', lastname: 'Wonderland' },
  { id: 2, firstname: 'Bob', lastname: 'Builder' },
  { id: 3, firstname: 'Mallory', lastname: 'Malicious' },
]

const middleware = {
  Student: {
    fullname: (resolve, parent, args, context, resolveInfo) => {
      const result = resolve(parent, args, context, resolveInfo)
      console.log('Student.fullname resolved:', result)
      return result
    }
  }
}

const resolvers = {
  Query: {
    student: (parent, args, context, resolveInfo) => {
      console.log('Query.student:', parent, args)
      return students.find(student => student.id === args.id)
    },
    allStudents: (parent, args, context, resolveInfo) => {
      console.log('Query.allStudents:', parent, args)
      return students.slice(0, args.limit || students.length)
    },
  },
  Student: {
    fullname: (parent, args, context, resolveInfo) => {
      console.log('Student.fullname:', parent, args)
      if (args.reverse) return [parent.lastname, parent.firstname].join(', ')
      return [parent.firstname, parent.lastname].join(' ')
    }
  }
}

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

Abbildung 6: Apollo server implementation

Abbildung 7: Romeo and Juliet at the beginning of the story

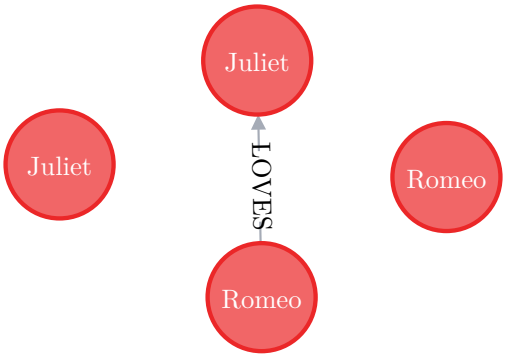

Abbildung 8: Oops, we have two Romeos and two Juliets now

7. VueJS and NuxtJS

VueJS: component composition pattern

VueJS: data bindings and events

VueJS: computed properties

VueJS: watchers

Nuxt: asyncData

Nuxt: page components and routing

SSR: JS globals e.g. ‘document‘ or ‘window‘

[4] (a) Zu den Vorteilen von client-seitig gerenderten “Single-page-applications” gegenüber traditionellen server-seitig gerenderten Web-Anwendungen gehören unter anderem:

- ◯ Weniger Last auf dem Server
- ◯ Desktop-ähnliches Nutzererlebnis
- ◯ Größerer Anteil Nutzdaten in der Kommunikation zwischen Client- und Server
- ◯ Bessere Suchmaschinen-Optimierung
- ◯ Kompatibilität mit älteren Browsern

[1] (b) Was ist isomorpher Quellcode?

.....

.....

[3] (c) Welche dieser sog. “JavaScript globals” sollte in Quellcode vermieden werden, welcher auch von server-seitig vorgerenderten Web-Anwendungen ausgeführt wird:

- ◯ `document`
- ◯ `console`
- ◯ `window`

[1] (d) Zu welcher Art von Fehler kommt es, wenn man trotzdem einer dieser “globals” in einem NodeJS Prozess aufruft?

.....

(e) Wirf einen Blick auf die Vue-Komponente im Code-Beispiel 9. Wenn in den folgenden Teilaufgaben Änderung im Quelltext vorgenommen werden sollen, können diese direkt in die Abbildung eingepflegt werden.

[1] i. Offenbar hat sich ein Fehler eingeschlichen. Eigentlich soll die Methode `handleClick` beim Klick auf den Knopf ausgelöst werden. Durch welche Änderung kann der Fehler behoben werden?

[1] ii. Wenn die Methode `handleClick` ausgelöst wird, welchen Wert hat dann `message`?
Antwort: _____

[2] iii. Füge ein `<input>` Feld hinzu, dessen Texteingabe mit dem Zustand `message` verknüpft ist. Bei jeder Text-Eingabe soll sich also der Wert von `message` verändern.

[2] iv. Implementiere eine computed property namens `reversedMessage`, welche den String `message` umdreht. Tipp:

```
someString.split('').reverse().join('')
```

Dreht eine lokale Variable `someString` herum.

[1] v. Warum ist es eigentlich sinnvoll, `reversedMessage` als computed property zu implementieren?

.....

.....

[1] vi. Was spricht dagegen, `reversedMessage` als Zustand in `data` zu initialisieren und zur Laufzeit zu verändern?

.....

.....

[1] vii. Was spricht dagegen, es als Methode zu implementieren und als `reversedMessage()` im "template" aufzurufen?

.....

.....

```
<template>
  <div>
    <!-- Hier kann das <input> Feld eingefügt werden -->


    <p>
      {{ message }}
    </p>
    <button>
      Click me
    </button>
    <p>
      {{ reversedMessage }}
    </p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      message: 'Hello!'
    }
  },
  computed: {
    // Hier kann die computed property namens `reversedMessage` implementiert
    // werden



  },
  methods: {
    handleClick() {
      this.message = `${this.message}!`
    },
  },
}
</script>
```

Abbildung 9: Eine kleine VueJS-Komponente

8. Fullstack testing and Requirements Engineering

Benefits and drawbacks of fullstack testing

Common pitfalls of fullstack testing, e.g. timeouts and checking non-existing HTML elements

User story template - who, what, why?

Cucumber scenarios and readability

1 (a) Im Kurs haben wir die "Testing-Pyramide" und den "Testing-Diamanten" kennengelernt. Beides sind Modelle, die eine Hilfestellung bieten sollen, um zu einem ausgewogenen Verhältnis der jeweiligen Test-Klassen zu finden. Welche Klasse von Tests hat laut dem "Testing-Diamanten" den größten Wert und sollte deshalb besonders oft implementiert werden?

Antwort: _____

2 (b) Für welchen Zweck nutzt man Cucumber Testing?

.....

.....

2 (c) Die Schablone eine User-Story sieht wie folgt aus:

```
As a <role>
I want to do <feature>
In order to <benefit>
```

Diese Schablone soll dazu dienen, dass man bei der Erfassung von Anforderungen bestimmte Informationen nicht vergisst, z.B. die Zielgruppe und den Grund für eine gewünschte Funktion. Warum ist es so wichtig, diese Informationen stets zu dokumentieren?

.....

.....

.....

.....

9. CSS

```
1 <form class="login-form">
2   <input type="text">
3   <input type="password">
4   <button class="submit-button">
5     <span>Login</span>
6     <span>- please</span>
7   </button>
8 </form>
```

Abbildung 10: Ein beispielhaftes HTML Dokument

3 (a) Welche HTML-Elemente werden bzw. welches HTML-Element wird in Abbildung 10 mit den folgenden CSS Selektoren selektiert? Gib als Antwort Zeilennummern an:

i.
```
.login-form:first-child
```
Antwort: _____

ii.
```
.login-form :first-child
```
Antwort: _____

iii.
```
.login-form > :first-child
```
Antwort: _____

1 (b) Mit welchen CSS-Selektor können alle Eingabefelder selektiert werden?

Antwort: _____

10. Code Review

I will use a VueJS component or an apollo server test as a code example for the 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.

Non-atomic test cases

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 (e.g. API keys or SSH keys) checked into source code

Not waiting for a promise to finish - tests pass before even checking a result

Nested if-clauses instead of guard clauses

Expose user listing in login resolver - slight security issue

Clients cannot update their cache becaus types have no ID in type definitions

Not using hard-coded values for expectations - danger of false negatives