Projekt: U-Car

Projektplan

1 Dokumenteninformation

1.1 Änderungsgeschichte

Datum	**Version**	**Änderungsgrund**	**Autorin**
24.03.23	1.0	Erstellung	Wenzl
29.03.23	1.1	Aktualisierung Arbeitspakete, Meilensteine, Arbeitsergebnisse	Wenzl
11.04.23	1.2	Ergänzung Unterstützende Prozesse	Wenzl
12.04.23	1.3	Hinzufügen von Verantwortlichkeiten	Wenzl
12.04.23	1.4	Aktualisierung Prototypfunktionen	Kronberg
20.04.25	1.5	Aktualisierung UC fully dressed / Wartebarkeit	Kenmo

1.2 Inhalt

2 Einführung

2.1 Definitionen und Abkürzungen

GUI: Graphic User Interface

2.2 Referenzen

3 Projektübersicht

Im Rahmen dieses Projekts soll eine Anwendung für Car-Sharing namens „U-Car“ entwickelt werden. Fahrzeugeigentümer können ihre Fahrzeuge registrieren und für bestimmte Zeiträume zur Buchung freigeben. Andere Nutzer können eine Reservierung oder Buchung anfragen, die durch den Eigentümer bestätigt werden muss. Dadurch können sich die Nutzer für ihren gewünschten Zeitraum und gegen eine Gebühr ein Auto ausleihen, während die Fahrzeugeigentümer am Monatsende eine Gutschrift je nach Entleihdauer erhalten.

3.1 Zweck und Ziel

Die Anwendung „U-Car“ soll es Privatpersonen ermöglichen, ihre wenig genutzten Autos zu vermieten oder ein Auto zu buchen, falls sie selbst keines besitzen.

3.2 Annahmen und Einschränkungen

Die Speicherung der Nutzer- und Fahrzeugdaten erfordert eine Schnittstelle zu einer MySQL-Datenbank, ohne die unsere Anwendung nicht funktionsfähig ist.

3.3 Arbeitsergebnisse

Arbeitsergebnis	**Beschreibung**
U-Car	fertige, funktionsfähige Anwendung
Quellcode	Dokumentation des gesamten Quellcodes der Anwendung
Dokumentation	Projektplan, Anforderungsspezifikationen, Domänenmodell, Softwareentwurf mit Klassendiagrammen, Testplan und -dokumentation, Abschlusspräsentation
Website	Informationen zum Projekt
Zeiterfassung	Dokumentation der benötigten Arbeitszeit

Die Tabelle wird im weiteren Projektverlauf aktualisiert.

4 Projektorganisation

4.1 Organisationsstruktur

Unser Team besteht aus folgenden Mitgliedern:

Mitglied	**Verantwortlichkeit**
Sandra Edigin	*Virtuelle Maschine*
Rosine Kenmo	
Corinna Kronberg	*Datenbank*
Enya Wenzl	*GitLab, Projektautomatisierung*

Die Tabelle wird im weiteren Projektverlauf aktualisiert.

4.2 Externe Schnittstellen

- MySQL-Datenbank
- GitLab-Repository

5 Management-Abläufe

5.1 Ressourceneinschätzungen

Für die Fertigstellung des Projekts stehen 15 Wochen und 150 Arbeitsstunden pro Person zur Verfügung, was einen Gesamtaufwand von 600 Arbeitsstunden und eine wöchentliche Arbeitszeit von 10 Stunden für jeden ergibt.

5.2 Projektplan

5.2.1 Zeitplan (Interne Meilensteine)

Meilenstein	**Frist**	**Beschreibung**
I1	29.03.23	• Entwurf von Use Cases, Domänenmodell, Projektplan
I2	12.04.23	• Fertigstellung von I1
I3	19.04.23	• Entwurf Architektur, Klassendiagramme, Datenbank, Testplan
I4	03.05.23	• Fertigstellung von I3 • Implementierung der grundlegenden Klassen und Architektur • Einbindung der Datenbank
I5	17.05.23	• Code Review jedes Teammitglieds
I6	24.05.23	• Fertigstellung von I5 • Fertigstellung des Prototyps
I7	14.06.23	• Fertigstellung der Anwendung • Durchführung aller nötigen Tests

I8	23.06.23	• Abschlusspräsentation, Website • Abgabe des Projekts

Die Tabelle wird im weiteren Projektverlauf aktualisiert.

5.2.2 Besprechungen

Jede Woche ist eine Besprechung geplant, die mittwochs um 8:15 Uhr an der OTH Regensburg stattfinden soll und an der alle Teammitglieder teilnehmen sollen. Dabei werden die Ergebnisse der letzten Woche und der Projektfortschritt besprochen sowie Aufgaben für die nächste Woche verteilt. Bei Bedarf können auch weitere Besprechungen zu anderen Zeiten oder mit einzelnen Mitgliedern abgehalten werden.

5.2.3 Abgaben

Meilenstein	Typ	Frist	Beschreibung
MS1	Dokumentation	21.04.23	Projektplan, Anforderungen, Domänenmodell
MS2	Dokumentation, Prototyp	26.05.23	Architektur, Entwurf, Prototyp
MS3	Dokumentation, Anwendung	09.07.23	Schlussabgabe des gesamten Projekts

6 Risiko Management

Risiko	Gegenmaßnahmen
Datenbankanbindung	frühzeitige Ausarbeitung der Schnittstelle

7 Arbeitspakete

Arbeitspaket	Inhalt	Verantwortliche	Zeit (h)	Abhängigkeit
Meetings		alle	100	
Projektplanung	Vision-Dokument, Projektplan	alle	16	
Domänen-modellierung	Domänenmodell, SSD, Contracts	Edigin, Wenzl	30	
Anforderungen	funktional und nicht-funktional, Use Cases	Kenmo, Kronberg	20	
Softwareentwurf			70	
Test			50	
Code Review			25	

Automatisierung			20	
Datenbank			30	
Einbindung der Datenbank			50	
GUI			70	
Domain Logic			90	
Versionskontrolle			5	
Website			8	
Abschluss-dokumentation			8	
Zeiterfassung		Kronberg	1	
		Gesamt:	593	
		Puffer:	7	

Die Tabelle wird im weiteren Projektverlauf aktualisiert.

7.1 Prototyp

Für den zweiten Meilenstein ist die Abgabe eines Prototyps erforderlich, der die folgenden Funktionen umfassen soll:

- Grundlegende Implementierung und Umsetzung der Schichtenarchitektur
- Kommunikation zwischen den Schichten
- Vollständige Konfiguration der Datenbank
- GUI mit grundlegenden Funktionen für Benutzerinteraktion
- Implementierung aller im Klassendiagramm genannten Klassen
- Anmelden und registrieren der Nutzer
- Fahrzeug registrieren
- Fahrzeug suchen und buchen
- Fahrzeug abholen und zurückgeben mit Erstellung der Rechnung

Dabei haben wir uns für den Prototypen bewusst gegen diese Funktionen entschieden:

- GUI vollständig ausgebaut und nicht nur funktionale Anforderungen erfüllt, sondern auch auf nicht funktionale Anforderungen (wie Benutzerfreundlichkeit) erweitert
- Nutzer löschen
- Administrator Funktionen
- Buchung stornieren
- Fahrzeug löschen

8 Infrastruktur

Einsatzzweck	**Infrastruktur**
Entwicklungsumgebung	QtCreator, Visual Studio Code, CLion

Datenbank	MySQL
Versionskontrolle, Fehlerbehandlung	GitLab
Test	googletest
Projektautomatisierung	CMake
Dokumentation	MS Office, www.diagrams.net
weitere	Virtuelle Maschine

Die Tabelle wird im weiteren Projektverlauf aktualisiert.

9 Unterstützende Prozesse

9.1 Konfigurationsmanagement

Für das Konfigurationsmanagement verwenden wir das bereitgestellte GitLab-Repository, um Quellcode und Dokumente an einer Stelle zu sammeln und auszutauschen. Dokumente werden in den Ordner Documentation auf dem Branch main hochgeladen und die durchgeführten Änderungen in die Commit Message geschrieben. Was Teil eines anderen Dokuments ist oder später in eines eingefügt wird, wird zum entsprechenden Zeitpunkt aus dem Ordner gelöscht.

Unser Quellcode befindet sich im Branch develop, in dem für jede neue Funktion einzelne feature Branches erstellt werden, in die Code in Bearbeitung hochgeladen wird. Diese werden mit develop gemerged, wenn sie funktionsfähig sind und eine Code Review stattgefunden hat.

9.2 Fehlermanagement

Für die Fehlerbehandlung verwenden wir ebenfalls GitLab, damit gefundene Fehler direkt am geschriebenen Code dokumentiert und bewertet werden können. Anschließend werden diese zeitnah behoben, was ebenfalls dokumentiert wird, um die Funktionsfähigkeit unserer Anwendung zu gewährleisten. Dafür gibt es im Ordner Documentation eine Exceltabelle, in der die gefundenen Fehler mit dem entsprechenden Dateinamen und den Codezeilen sowie dem Behebungsstatus festgehalten werden.

9.3 Qualitätsmanagement

Geschriebener Code wird mindestens an den Stellen mit Kommentaren versehen, an denen seine Funktion nicht selbsterklärend ist. Durch regelmäßige Code Reviews wird die Qualität unseres Codes sichergestellt, außerdem werden die Funktionsfähigkeit der einzelnen Komponenten und ihre Kompatibilität ausreichend getestet.

9.4 Zeiterfassung

Die Zeiterfassung erfolgt mittels einer gemeinsamen Exceltabelle, in der die Arbeitszeit jedes Teammitglieds für die einzelnen Arbeitspakete festgehalten wird.

Projekt: U-Car

Anforderungsspezifikationen

1 Dokumenteninformation

1.1 Änderungsgeschichte

Datum	**Version**	**Änderungsgrund**	**Autorin**
26.03.23	1.0	Erstellung	Wenzl
01.04.23	1.1	Ergänzung nichtfunktionaler Anforderungen	Wenzl
13.04.23	1.2	Ergänzung funktionaler Anforderungen, Use Cases und Use Case Diagramm	Kronberg
20.04.23	1.3	Aktualisierung UC fully dressed / Wartebarkeit	Kenmo

1.2 Inhalt

2 Einführung

2.1 Definitionen und Abkürzungen

Nutzer bezeichnet die Person ohne Berechtigungen.

Fahrzeugeigentümer bezeichnet die Person, die berechtigt ist, Fahrzeuge zu registrieren.

Wenn beide zusammen auftreten, stellt der Nutzer den Mieter und der Fahrzeugeigentümer den Vermieter dar, unabhängig von den anderen Berechtigungen des Nutzers.

2.2 Referenzen

3 Allgemeine Beschreibung

3.1 Produktperspektive

U-Car ist eine Anwendung für Privatpersonen, die ihr Autos nicht oft nutzen oder ein Auto benötigen, aber keines besitzen. Fahrzeugeigentümer können ihre Fahrzeuge registrieren und für bestimmte Zeiträume zur Buchung freigeben. Andere Nutzer können eine Reservierung oder Buchung anfragen, die durch den Eigentümer bestätigt werden muss. Dadurch können sich die Nutzer für ihren gewünschten Zeitraum und gegen eine Gebühr ein Auto ausleihen, während die Fahrzeugeigentümer am Monatsende eine Gutschrift je nach Entleihdauer erhalten.

3.2 Produktfunktion

- Anmeldung mit verschiedenen Berechtigungen für Privatpersonen, Fahrzeugeigentümer und Administratoren
- Ausgeben einer Liste von verfügbaren Fahrzeugen und Buchung eines Fahrzeugs
- Bestätigung von Buchungen durch Fahrzeugeigentümer
- Nutzerkonto mit Profilinformationen und Zahlungsdetails
- Registrierung von Fahrzeugen durch die Eigentümer
- Fahrzeugübergabe- und -rückgabeoptionen, einschließlich Standortinformationen und Zustandsberichten
- Ausstellen einer Rechnung für jede Buchung

3.3 Benutzercharakteristik

Zielgruppe des Produkts sind einerseits Menschen, die ihr eigenes Auto nur selten nutzen und deswegen nach einer Möglichkeit suchen, es an andere zu vermieten. Andererseits richtet sich U-Car aber auch an diejenigen, die selbst kein Auto besitzen und deshalb nur für eine bestimmte Zeit ein Fahrzeug mieten wollen.

3.4 Einschränkungen

Benachrichtigungen, die eigentlich per E-Mail an die Nutzer verschickt werden, werden stattdessen in ein Textdokument geschrieben.

3.5 Annahmen

Die vom Nutzer eingegebenen Daten werden als korrekt angesehen.

3.6 Abhängigkeiten

Unsere Anwendung ist von der funktionsfähigen Anbindung einer Datenbank abhängig.

3.7 Use Case Überblick

- UC1: Konto registrieren/anmelden
- UC2: Fahrzeug registrieren
- UC3: Fahrzeug suchen
- UC4: Fahrzeug buchen
- UC5: Fahrzeug abholen und zurückbringen
- UC6: Buchung stornieren

4 Spezifische Anforderungen

4.1 Funktionale Anforderungen

4.1.1 Funktionale Anforderung F1 – Kontoregistrierung und -anmeldung

Die Nutzer sollen sich registrieren/ anmelden können, um Zugang zum Carsharing-Dienst zu erhalten. Bei der Registrierung sollen sie ihre persönlichen Daten wie Name, Adresse, Geburtsdatum, E-Mail-Adresse und Passwort angeben können. Bei der Anmeldung sollen sie ihren Namen und ihr Passwort angeben können.

4.1.2 Funktionale Anforderung F2 – Fahrzeugregistrierung

Der Fahrzeugeigentümer soll Fahrzeuge in seinem Profil registrieren können. Dabei soll er einige Angaben zu seinem Fahrzeug machen, wie z.B. das Fahrzeugmodell, die Anzahl der Sitzplätze und die Adresse. Bei der Adressauswahl soll das System bereits bestehende Stellplatzadressen anzeigen, um dem Fahrzeugeigentümer die Möglichkeit zu geben, aus diesen auszuwählen. Nach Bestätigung durch den Fahrzeugeigentümer soll das System das Fahrzeug und ggf. den neuen Stellplatz anlegen.

4.1.3 Funktionale Anforderung F3 – Fahrzeugsuche

Die Nutzer sollen die im System gespeicherten verfügbaren Fahrzeuge suchen können. Sie sollen den gewünschten Standort, den Fahrzeugtyp und das Datum angeben können.

4.1.4 Funktionale Anforderung F4 – Fahrzeugbuchung

Nachdem der Nutzer das gewünschte Fahrzeug gefunden hat, soll er dieses buchen können. Das System sollte dann den Fahrzeugeigentümer informieren und auf dessen Bestätigung warten, bevor das System das Fahrzeug endgültig bucht.

4.1.5 Funktionale Anforderung F5 – Fahrzeugabholung und -rückgabe

Bei der Fahrzeugübernahme soll der Nutzer seine Startzeit und den aktuellen Kilometerstand ins System eingeben. Zusätzlich soll er die Möglichkeit zur Eingabe spezifischer Informationen, wie z.B. Tankfüllung oder Schäden, haben. Nach der Nutzung soll es der Nutzer die Endzeit, der Kilometerstand und ggf. weitere Informationen vom Nutzer in das System eingegeben können.

4.1.6 Funktionale Anforderung F6 – Rechnung

Nach jeder Nutzung sollen der Nutzer und der Fahrzeugeigentümer automatisch eine Rechnung erhalten. Diese soll Angaben zum Nutzer, zum Fahrzeugeigentümer, zum Fahrzeugtyp, zum konkreten Fahrzeug und zu den gefahrenen Kilometern enthalten.

4.1.7 Funktionale Anforderung F7 – Stornierung

Der Nutzer soll die Möglichkeit haben, seine aktuell vorgenommenen Buchungen einschließlich Buchungszeitraum und Fahrzeugtyp einzusehen und zu stornieren. Der Fahrzeugeigentümer soll eine Benachrichtigung über die geänderte Buchung erhalten.

4.1.8 Funktionale Anforderung F8 – Administratorkonto

Das System soll in der Lage sein, dem Administrator eine Übersicht mit allen Informationen über die bisher erfolgten Buchungen, Nutzer, Fahrzeuge und Stellplätze darzustellen.

4.2 Bedienbarkeit

4.2.1 Bedienbarkeitsanforderung 1

Die Anwendung soll so gestaltet sein, dass sie bedienbar ist und alle Funktionen für den Nutzer ersichtlich sind. Außerdem alten Personen mit wenig oder niedrigen IT-Kenntnissen soll das System ohne Hilfe in wenige Minuten bedienen können.

4.2.2 Bedienbarkeitsanforderung 2

Die Anwendung soll mit Maus und Tastatur bedient werden.

4.3 Zuverlässigkeit

4.3.1 Zuverlässigkeitsanforderung 1

Durch regelmäßige Tests soll die Funktionsfähigkeit der Anwendung gewährleistet werden.

4.3.2 Zuverlässigkeitsanforderung 2

Beim Ausstellen einer Rechnung soll der Betrag in jedem Fall korrekt berechnet werden, sofern die Daten zu Fahrzeug und Fahrt korrekt eingegeben wurden.

4.4 Leistung

4.4.1 Leistungsanforderung 1

Das System soll nicht länger als drei Sekunden brauchen, um zu einer anderen Seite zu wechseln. Datenbankanfragen richten sich nach der Größe der Datenbank.

4.5 Wartbarkeit

4.5.1 Wartbarkeitsanforderung 1

Die registrierten Nutzer, Fahrzeuge und Buchungen sollen über ein Administratorkonto verwaltet werden können. Außerdem sollen die Anwendung später noch warten oder verändern können.

4.6 Sicherheit

4.6.1 Sicherheitsanforderung 1

personenbezogenen Daten sollen nur mit Einwilligung gearbeitet werden.

4.6.2 Sicherheitsanforderung 2

Für alle Nutzerkonten soll das Berechtigungslevel vermerkt werden, um die Verteilung der Administratorrechte zu sichern.

4.7 Schnittstellen

4.7.1 Benutzerschnittstelle

Der Benutzer soll die Anwendung am eigenen Rechner verwenden und über ein grafisches Interface steuern können.

4.7.2 Datenbankschnittstelle

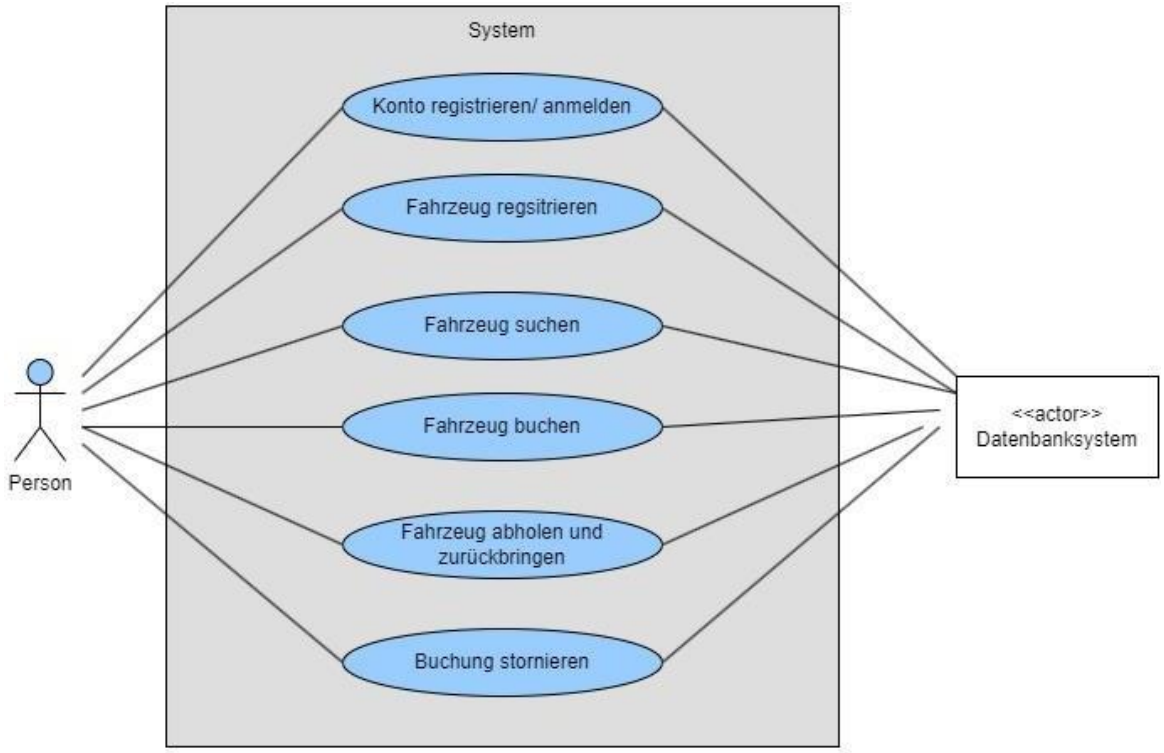
Zur Speicherung der Nutzer- und Fahrzeugdaten soll eine MySQL-Datenbank verwendet werden.

4.8 Installation

Die Anwendung soll nach der Fertigstellung zum Download für Linux bereitstehen.

5 Use Cases

5.1 Use Case – Diagramm



5.2 Use Case UC1: Konto registrieren/anmelden

Primary Actor: Nutzer

Stakeholders and Interests:

- Nutzer: möchte ein neues Konto erstellen/ sich anmelden
- Admin: will, dass die Registrierung oder Anmeldung erfolgreich läuft

Preconditions: Der Benutzer hat U-Car installiert.

Postconditions: Der Benutzer hat ein neues Konto erstellt/ ist nun in U-Car eingeloggt.

Main Success Scenario:

1. Der Benutzer gibt seine Daten (Name und Passwort/ Name, Adresse, Geburtsdatum, E-Mail-Adresse und Passwort) an.
2. Das System überprüft die eingegebenen Informationen auf ihre Richtigkeit und Validität.

3. Bei der Registrierung erstellt das System ein neues Konto für den Benutzer.
4. Das System leitet den Benutzer auf die Hauptseite der Carsharing-Software weiter.

Extensions:

2a. Die Daten sind nicht korrekt.

1. Das System benachrichtigt den Nutzer, dass die Daten nicht korrekt sind.
weiter mit 1.

Special Requirements:

Technology und Data Variations List:

Frequency of Occurrence: mehrmals täglich

Open Issues:

5.3 Use Case UC2: Fahrzeug registrieren

Primary Actor: Fahrzeugeigentümer

Stakeholders and Interests:

- Fahrzeugeigentümer: will, dass sein Fahrzeug erfolgreich registriert wurde und alle Angaben stimmen
- Admin: möchte, dass das Fahrzeug und alle Informationen und im Fahrzeugkatalog gespeichert werden

Preconditions: Der Nutzer hat die Benachrichtigung des Fahrzeugeigentümer und ist angemeldet.

Postconditions: Das neue Fahrzeug und seine Daten wurden gespeichert und es wurde dem Fahrzeugkatalog hinzugefügt.

Main Success Scenario:

1. Der Fahrzeugeigentümer gibt Angaben zu seinem Fahrzeug, so wie Fahrzeugmodell, Kennzeichen, Farbe, Anzahlsitzplätze, mögliche Mietzeiträume, Anzahl der Sitzplätze an.
2. Das System zeigt alle bisher gespeicherten Stellplätze an.
3. Der Fahrzeugeigentümer wählt einen dieser Stellplätze aus.
4. Der Fahrzeugeigentümer gibt die Stellplatznummer ein und bestätigt alle Eingaben.
5. Das System speichert das neue Fahrzeug und alle weiteren angegebenen Informationen und fügt es dem Fahrzeugkatalog hinzu.

Extensions:

3a. Der Fahrzeugeigentümer will keinen dieser Stellplätze auswählen.

1. Der Fahrzeugeigentümer gibt eine andere Adresse ein.
2. Das System speichert die neue Adresse als Stellplatz.

Weiter bei 4.

Special Requirements:

Technology und Data Variations List:

Frequency of Occurrence: mehrmals täglich

Open Issues:

5.4 Use Case UC3: Fahrzeug suchen

Primary Actor: Nutzer

Stakeholders and Interests:

- Nutzer: will, dass alle verfügbaren Fahrzeuge zu seiner Suche angezeigt werden

Preconditions: Der Nutzer hat sich erfolgreich angemeldet.

Postconditions: Der Nutzer hat eine Liste der verfügbaren Fahrzeuge erhalten, die seinen Suchkriterien entsprechen.

Main Success Scenario:

1. Der Nutzer gibt seine gewünschten Daten für die Suchfunktion, wie z.B. den Standort, den Fahrzeugtyp und das Datum, ein.
2. Das System zeigt eine Liste von Fahrzeugen an, die den Suchkriterien des Nutzers entsprechen.
3. Der Nutzer wählt das gewünschte Fahrzeug aus.

Extensions:

*a. Der Nutzer bricht die Suche ab.

1. Das System zeigt das Suchfenster an.

2a. Es ist kein Fahrzeug mit den gewünschten Daten verfügbar.

1. Das System benachrichtigt den Nutzer und zeigt das Suchfenster an.

Special Requirements:

Technology und Data Variations List:

Frequency of Occurrence: Vor jeder Buchung, mehrmals pro Stunde

Open Issues:

5.5 Use Case UC4: Fahrzeug buchen

Primary Actor: Nutzer

Stakeholders and Interests:

- Nutzer: will, dass das Fahrzeug zu dem Mietzeitpunkt gebucht ist und die Buchung in seiner Buchungsliste vermerkt ist
- Fahrzeugeigentümer: will, dass das Fahrzeug für den angegeben Zeitpunkt gebucht ist und er alle erforderlichen Nutzerdaten erhalten hat
- Admin: will, dass die Buchung erfolgreich läuft

Preconditions: Der Nutzer hat ein verfügbares Fahrzeug gefunden.

Postconditions: Das Fahrzeug ist für den Nutzer für die gewünschten Zeit gebucht und steht zu dem gebuchten Zeitpunkt zur Verfügung. Die Buchung ist in der Buchungsliste des Nutzers und der Buchungsliste für das Fahrzeug aufgelistet.

Main Success Scenario:

1. Das System zeigt das vom Nutzer ausgewählte Fahrzeug an und fragt nach einer Bestätigung der Auswahl.
2. Der Nutzer bestätigt seine Auswahl.
3. Das System bestätigt die Buchung und zeigt die Details der Reservierung an.
4. Das Fahrzeug für den Nutzer reserviert und ist zu dem gebuchten Zeitpunkt verfügbar.
5. Das System benachrichtigt den Fahrzeugeigentümer und übermittelt ihm alle erforderlichen Nutzerdaten.
6. Der Fahrzeugeigentümer bestätigt die Buchung.
7. Das System informiert den Nutzer über die Bestätigung.

Extensions:

2a. Ein anderer Nutzer hat das gleiche Fahrzeug zur gleichen Zeit gebucht

1. Das System zeigt eine Fehlermeldung an und fordert den Nutzer auf, eine andere Buchung vorzunehmen.
2. Das System zeigt die Suchliste an.

5a. der Fahrzeugeigentümer lehnt ab

1. Der Nutzer wird benachrichtigt und das Fahrzeug wird für den Mietzeitraum als frei gespeichert.

Special Requirements:

Technology und Data Variations List:

Frequency of Occurrence: mehrmals pro Stunde

Open Issues:

5.6 Use Case UC5: Fahrzeug abholen und zurückbringen

Primary Actor: Nutzer

Stakeholders and Interests:

- Nutzer: will ein fehlerfreies System, das alle seine Anfragen schnell bearbeitet und in dem die Preise korrekt berechnet werden
- Fahrzeugeigentümer: will, dass alle Daten erfasst und die Preise korrekt errechnet werden
- Unternehmen: will das alle Daten bezüglich der Buchung korrekt gespeichert werden und dass das Fahrzeug danach wieder als frei angegeben wird und dass die Rechnung korrekt erstellt wurde und im System gespeichert wurde
- Bundesamt für Steuern: will Steuern von jeder Rechnung erhalten

Preconditions: Der angemeldete Nutzer hat ein Fahrzeug gebucht und die Buchung wurde vom Fahrzeugeigentümer bestätigt. Er befindet sich am Abholort.

Postconditions: Das System hat alle Daten bezüglich der Zeiten, des Kilometerstandes und der zusätzlichen Informationen gespeichert. Das Fahrzeug wird wieder als frei angezeigt, sofern nicht im Anschluss gebucht. Der Nutzer und der Fahrzeugeigentümer haben eine Rechnung erhalten.

Main Success Scenario:

1. Der Nutzer gibt die Startzeit und den aktuellen Kilometerstand ins System ein. Zusätzlich kann er spezifische Informationen, wie z.B. Tankfüllung oder Schäden ins System eingeben.
2. Das System speichert diese Informationen.
3. Nach der Nutzung gibt der Nutzer Endzeit, Kilometerstand und ggf. weitere Informationen ins System ein.
4. Das System speichert diese Informationen.
5. Das System erstellt eine Rechnung mit Informationen über den Nutzer, den Fahrzeugeigentümer, den Fahrzeugtyp, das konkrete Fahrzeug, die tatsächlich gebuchten Zeiten, die gefahrenen Kilometer und den Zahlungsinformationen.
6. Aus diesen Angaben berechnet das System den Preis und fügt dies ebenfalls der Rechnung hinzu.
7. Das System schickt diese Rechnung an den Nutzer und den Fahrzeugeigentümer.

Extensions:

Special Requirements:

Technology und Data Variations List:

Frequency of Occurrence: Bei jeder Buchung, mehrmals täglich

Open Issues:

- Was passiert, wenn die Buchungszeit überschritten wurde und ein anderer Nutzer das Fahrzeug für diese Zeit gebucht hatte?
- Was passiert, wenn der Nutzer einen Unfall hatte?
- Welche Auswirkungen in der Rechnung haben Fahrzeugschäden, die durch die Benutzung entstanden sind?

5.7 Use Case UC6: Buchung stornieren

Primary Actor: Nutzer

Stakeholders and Interests:

- Nutzer: will das die Buchung korrekt storniert wurde
- Fahrzeugeigentümer: will über die Änderung benachrichtigt werden und sein Fahrzeug soll danach als „frei“ gelten
- Unternehmen: will das alle Kunden zufrieden sind, indem alle Prozesse korrekt ablaufen

Preconditions: Der Nutzer hat ein Fahrzeug gebucht und die Buchung wurde vom Fahrzeugeigentümer akzeptiert. Die Buchungszeit hat noch nicht begonnen.

Postconditions: Die Buchung wurde gelöscht. Das Fahrzeug gilt in dem Buchungszeitraum als nicht gebucht.

Main Success Scenario:

1. Der Nutzer sieht eine Übersicht seiner Buchungen.
2. Der Nutzer wählt eine noch nicht eingetretene Buchung aus und wählt Buchung stornieren im System aus.
3. Das System löscht die Buchung mit allen Informationen, setzt das Fahrzeug in dem Buchungszeitraum als nicht gebucht und benachrichtigt den Fahrzeugeigentümer.

Extensions:

2a. Der Nutzer wählt eine Buchung zu einer bereits vergangen Zeit aus.

1. Das System weist Nutzer darauf hin, dass diese Buchung bereits geschehen ist. Zurück zu 1.

Special Requirements:

Technology und Data Variations List:

Frequency of Occurrence: bei jeder hunderten Buchung

Open Issues:

- Wie viele Stunden vor dem Buchungsbeginn, darf der Nutzer die Buchung noch stornieren?

Projekt: U-Car

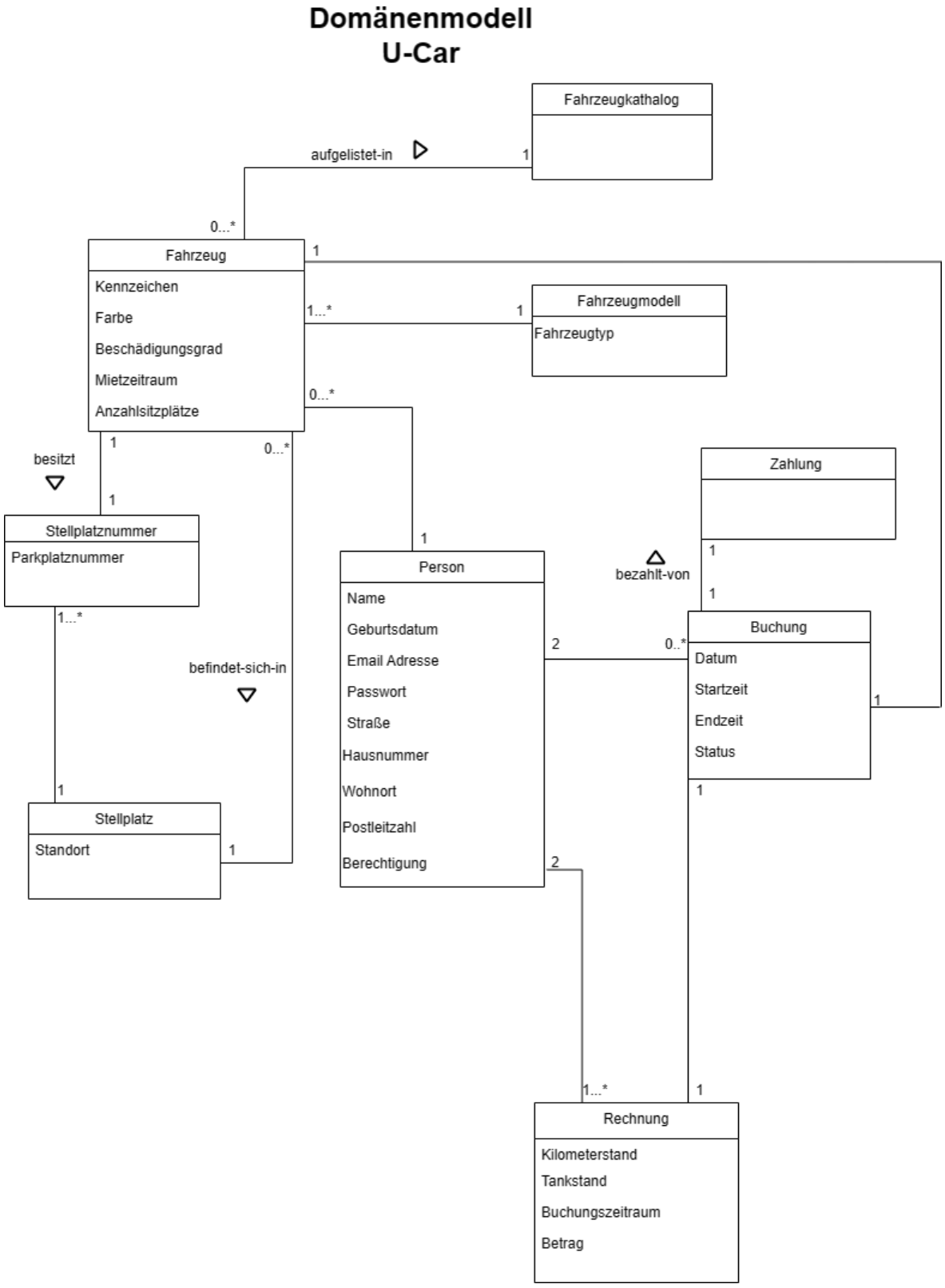
Domänenmodell

Domänenmodell

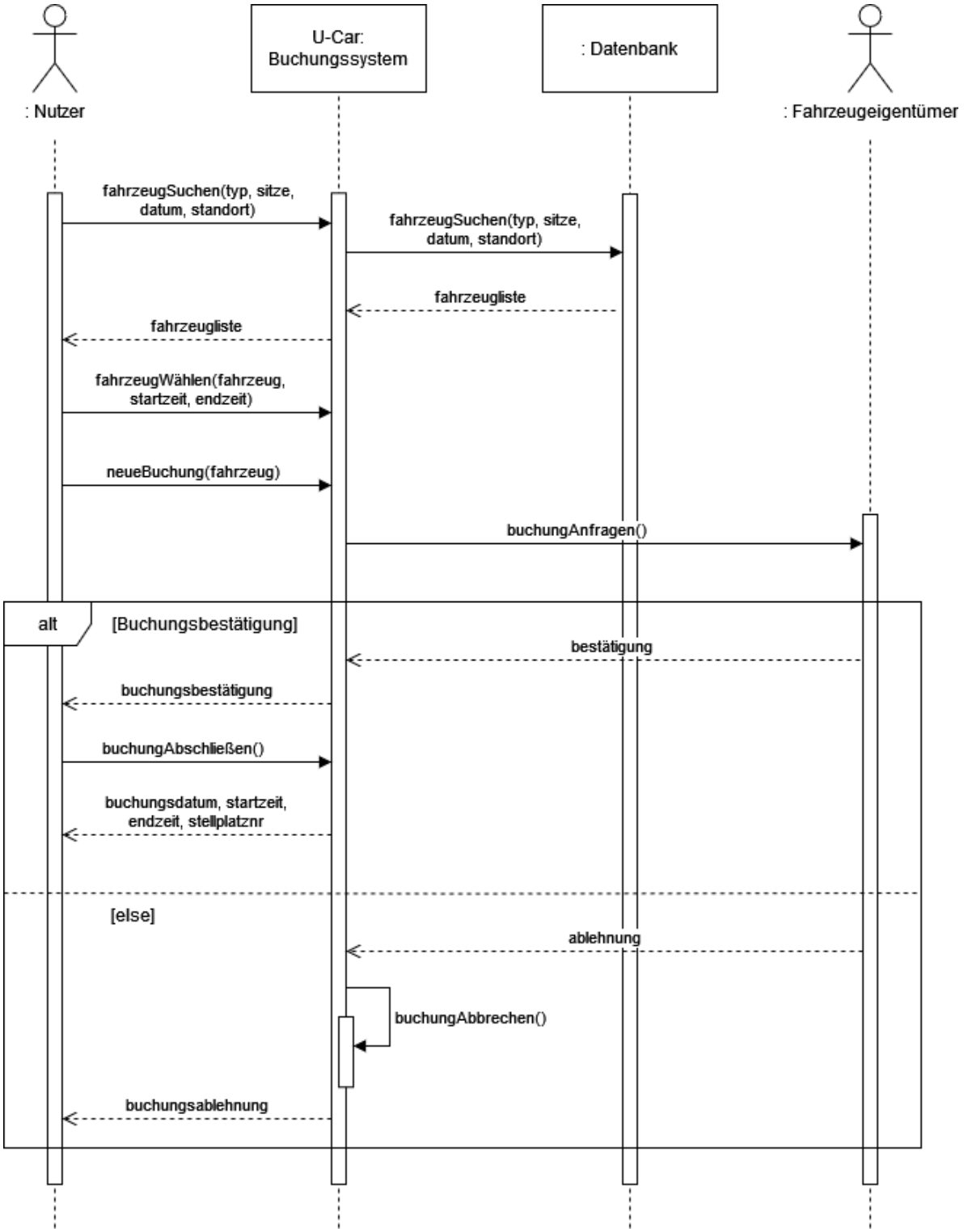
1 Änderungsgeschichte

Datum	**Version**	**Änderungsgrund**	**Autorin**
11.04.23	1.0	Zusammenfügen in ein Dokument	Wenzl
25.03.23	1.1	Erstellung des Domänenmodells	Edigin
16.04.2023	1.2	Änderung am Domänenmodell	Edigin

2 Domänenmodell



3 Systemsequenzdiagramm zu Use Cases 3 und 4



4 Contracts

operation contract	fahrzeugSuchen(typ, sitze, datum, standort)
crossreferences	UC3: Fahrzeug suchen
preconditions	Person angelegt mind. ein Fahrzeug mit Fahrzeugkatalog verknüpft
postconditions	

operation contract	fahrzeugWählen(fahrzeug, startzeit, endzeit)
crossreferences	UC4: Fahrzeug buchen
preconditions	Suche durchgeführt
postconditions	

operation contract	neueBuchung(fahrzeug)
crossreferences	UC4: Fahrzeug buchen
preconditions	Fahrzeug ausgewählt
postconditions	Buchung erzeugt Buchung mit Fahrzeug und Personen verknüpft Buchungseingang und Zeitraum aktualisiert Status der Buchung auf “in Bearbeitung” gesetzt

operation contract	buchungAbschließen()
crossreferences	UC4: Fahrzeug buchen
preconditions	Buchung angelegt Buchung mit Fahrzeug und Personen verknüpft
postconditions	Status der Buchung auf “bestätigt” gesetzt

operation contract	buchungAbbrechen()
crossreferences	UC4: Fahrzeug buchen
preconditions	Buchung abgelehnt
postconditions	Verknüpfung mit Fahrzeug und Personen entfernt Buchung gelöscht

Teilnahmeerklärung am Modul Software-Praktikum

Ich nehme am Software-Praktikum im Sommersemester 2023 teil.

Das Software - Praktikum wird als Projektarbeit durchgeführt und mit einem studienbegleitenden Leistungsnachweis (Portfolioprüfung zur Projektarbeit) bewertet.

Mir ist bekannt, dass nicht erbrachte Leistungen (u.a. infolge fehlender Teilnahme an einem Projektreview) mit der Note „nicht ausreichend" zu bewerten sind.

Regensburg, den 13.04.23

Enya Wenzl, 3313677 E. Wenzl

[Name, Matrikel - Nr.] [Unterschrift]

Teilnahmeerklärung am Modul
Software-Praktikum

Ich nehme am Software-Praktikum im Sommer semester 2023 teil.

Das Software – Praktikum wird als Projektarbeit durchgeführt und mit einem studienbegleitenden Leistungsnachweis (Portfolioprüfung zur Projektarbeit) bewertet.

Mir ist bekannt, dass nicht erbrachte Leistungen (u.a. infolge fehlender Teilnahme an einem Projektreview) mit der Note „nicht ausreichend“ zu bewerten sind.

Regensburg, den 14.04.2023

[Name, Matrikel – Nr.] [Unterschrift]

Edigin Sandra, 3222900

Teilnahmeerklärung am Modul
Software-Praktikum

Ich nehme am Software-Praktikum im Sommer Semester 2023 teil.

Das Software – Praktikum wird als Projektarbeit durchgeführt und mit einem studienbegleitenden Leistungsnachweis (Portfolioprüfung zur Projektarbeit) bewertet.

Mir ist bekannt, dass nicht erbrachte Leistungen (u.a. infolge fehlender Teilnahme an einem Projektreview) mit der Note „nicht ausreichend“ zu bewerten sind.

Regensburg, den 13.04.2023

Corinna Kronberg, 3320710 

[Name, Matrikel – Nr.] [Unterschrift]

Teilnahmeerklärung am Modul

Software-Praktikum

Ich nehme am Software-Praktikum (610100) im

SOSE 23 (Semester, Jahr)

teil.

Das Software – Praktikum wird als Projektarbeit durchgeführt und mit einem studienbegleitenden Leistungsnachweis bewertet.

Das Nichterscheinen zu einer Reviewveranstaltung oder zur Abschlussveranstaltung wird als Fehlleistung (Note 5) bewertet.

Regensburg, den 15 April 2023

[Name, Matrikel – Nr.] [Unterschrift]

Kenmo Tchouyola
Rosine Christine
3183232

Wichtiger Hinweis: Damit die Note elektronisch ohne Verzögerung im QIS eingetragen werden kann, melden Sie sich bitte auch dort für das Modul an!