Razvoj aplikacije za rentiranje automobila korišćenjem metodologije Design Thinking

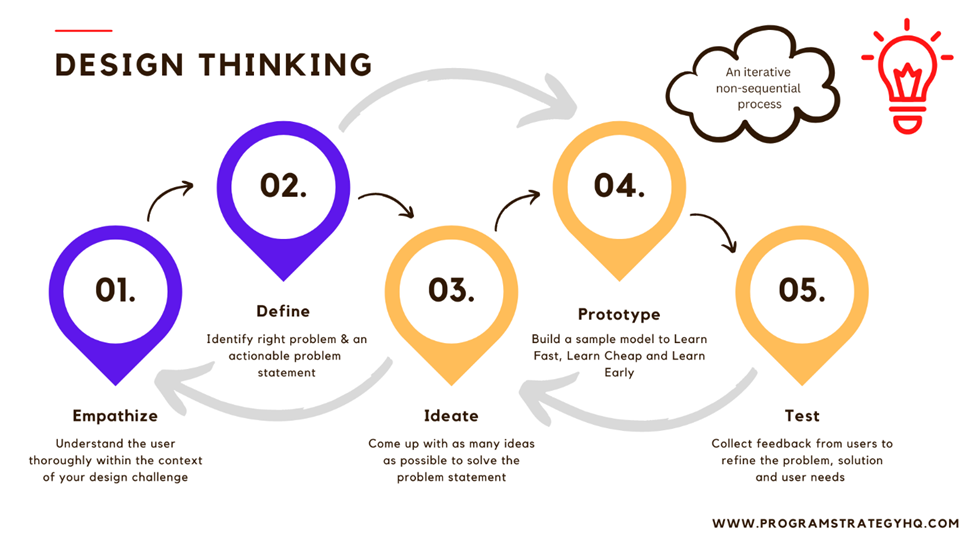
#### **Uvod**

Rentiranje automobila je postalo neizostavan deo modernog načina putovanja, ali proces često može biti komplikovan i neprilagođen korisnicima. Cilj ovog projekta je razvoj intuitivne i korisnički prijatne aplikacije za rentiranje automobila korišćenjem metodologije Design Thinking. Aplikacija omogućava korisnicima brzo pretraživanje, rezervaciju i upravljanje procesom rentiranja vozila putem jednostavnog i funkcionalnog interfejsa.



#### **Metodologija razvoja softvera**

U razvoju aplikacije primenjen je Design Thinking okvir, koji se fokusira na razumevanje potreba korisnika i iterativni pristup kreiranju rešenja. Ova metodologija obuhvata pet ključnih faza: Empatija, Definisanje problema, Generisanje ideja, Prototipovanje i Testiranje.



##### **2.1. Primena Design Thinking-a na projekat**

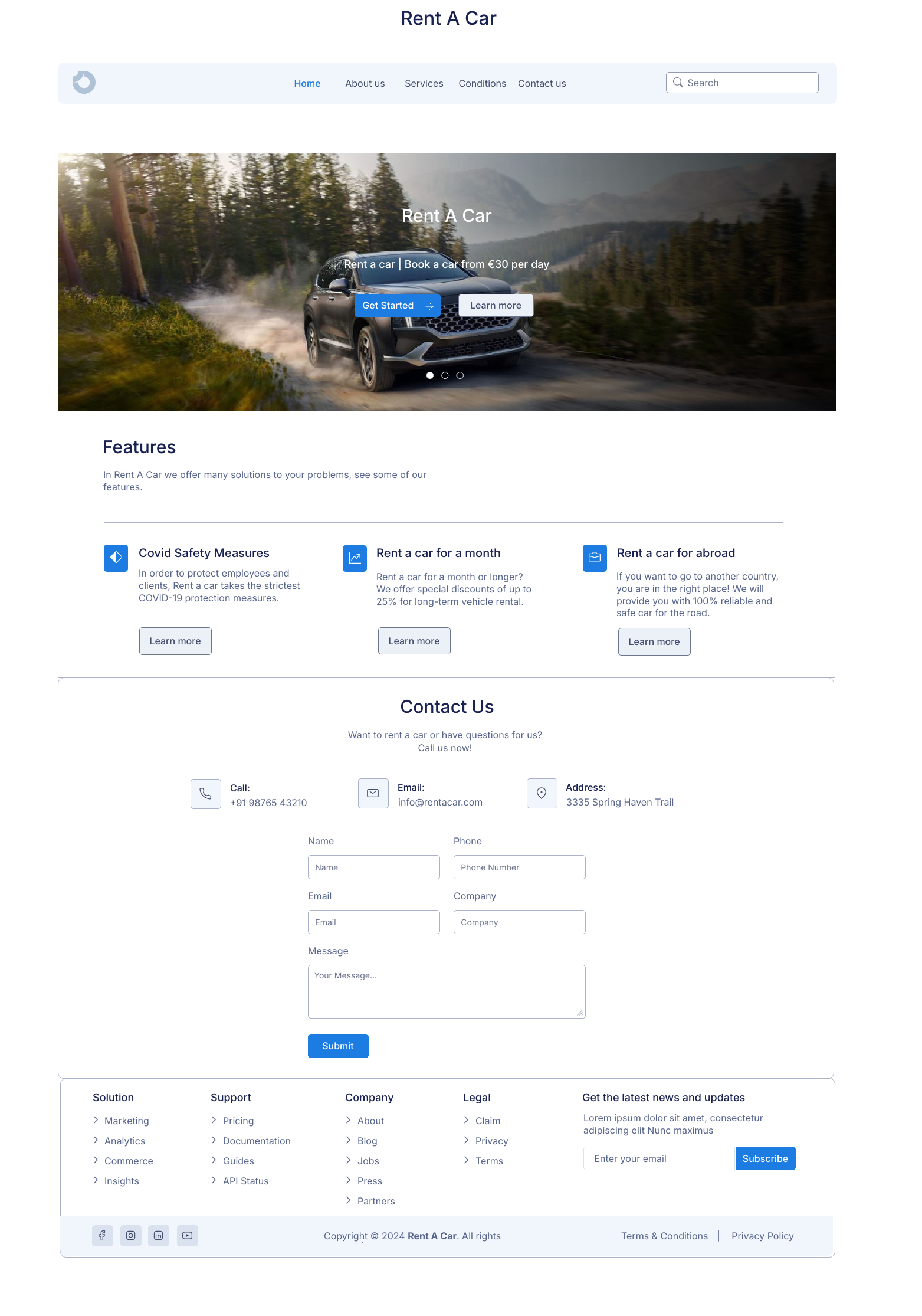
**Empatija:**  
Kroz ankete i intervjue prikupljeni su podaci o potrebama i bolnim tačkama korisnika prilikom rentiranja automobila. Na primer, korisnici su istakli da je komplikovan proces pretrage vozila i nejasne cene glavni problem.

**Definisanje problema:**  
Na osnovu povratnih informacija definisan je osnovni problem: "Kako omogućiti korisnicima jednostavan, brz i transparentan proces rentiranja automobila?"

**Generisanje ideja:**  
Tim je razvio niz ideja za rešavanje problema, uključujući funkcionalnosti poput filtera za pretragu (tip vozila, cena, lokacija), prikaza dostupnih vozila u realnom vremenu i transparentnog obračuna cena.

**Prototipovanje:**  
Kreiran je vizuelni prototip aplikacije koristeći alat Mockflow. Prototip sadrži glavne ekrane aplikacije:

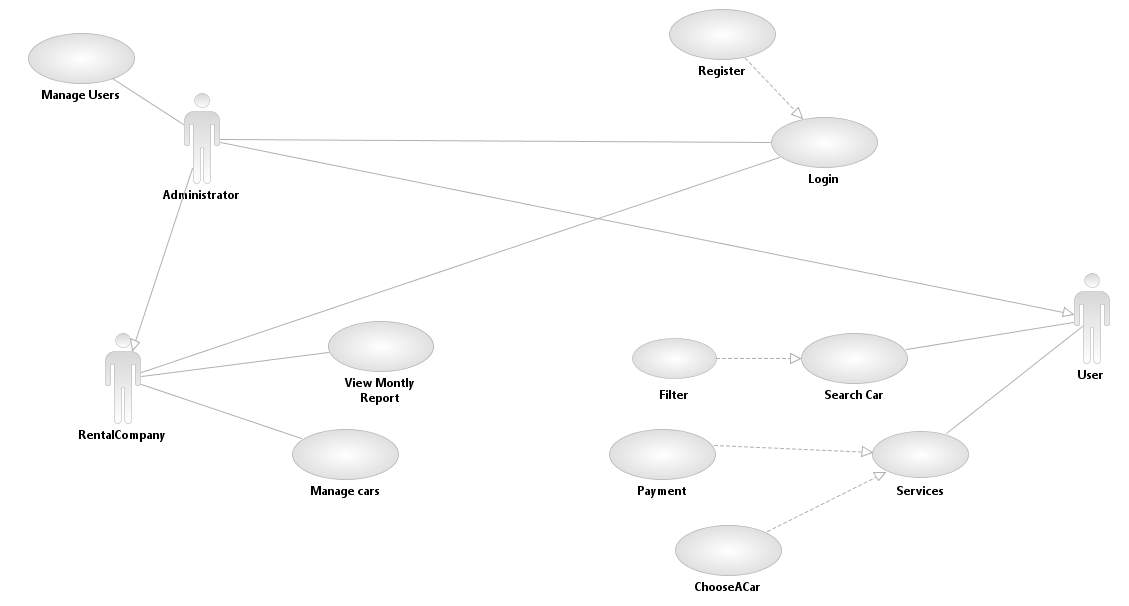
1. Početna stranica.
2. Servisi sa pretragom automobila.
3. Detalji o vozilu i proces rezervacije.
4. Pregled potvrde rezervacije.



**Testiranje:**  
Prototip je testiran sa grupom korisnika, a na osnovu njihovih povratnih informacija izvršene su izmene. Na primer, dodata je funkcionalnost brzog pregleda uslova rentiranja.

#### **Specifikacija korisničkih zahteva**

##### **3.1. Use Case Dijagram**



Ovaj dijagram predstavlja **use-case** model za sistem iznajmljivanja automobila. Prikazane su tri glavne uloge (aktera): **Administrator**, **Kompanija za iznajmljivanje** i **Korisnik**. Svaki akter ima određene funkcionalnosti koje može da izvršava unutar sistema:

#### **Administrator**

* Upravljanje korisnicima
* Nasleđuje sve ostale funkcionalnosti

#### **Kompanija za iznajmljivanje (Rental Company)**

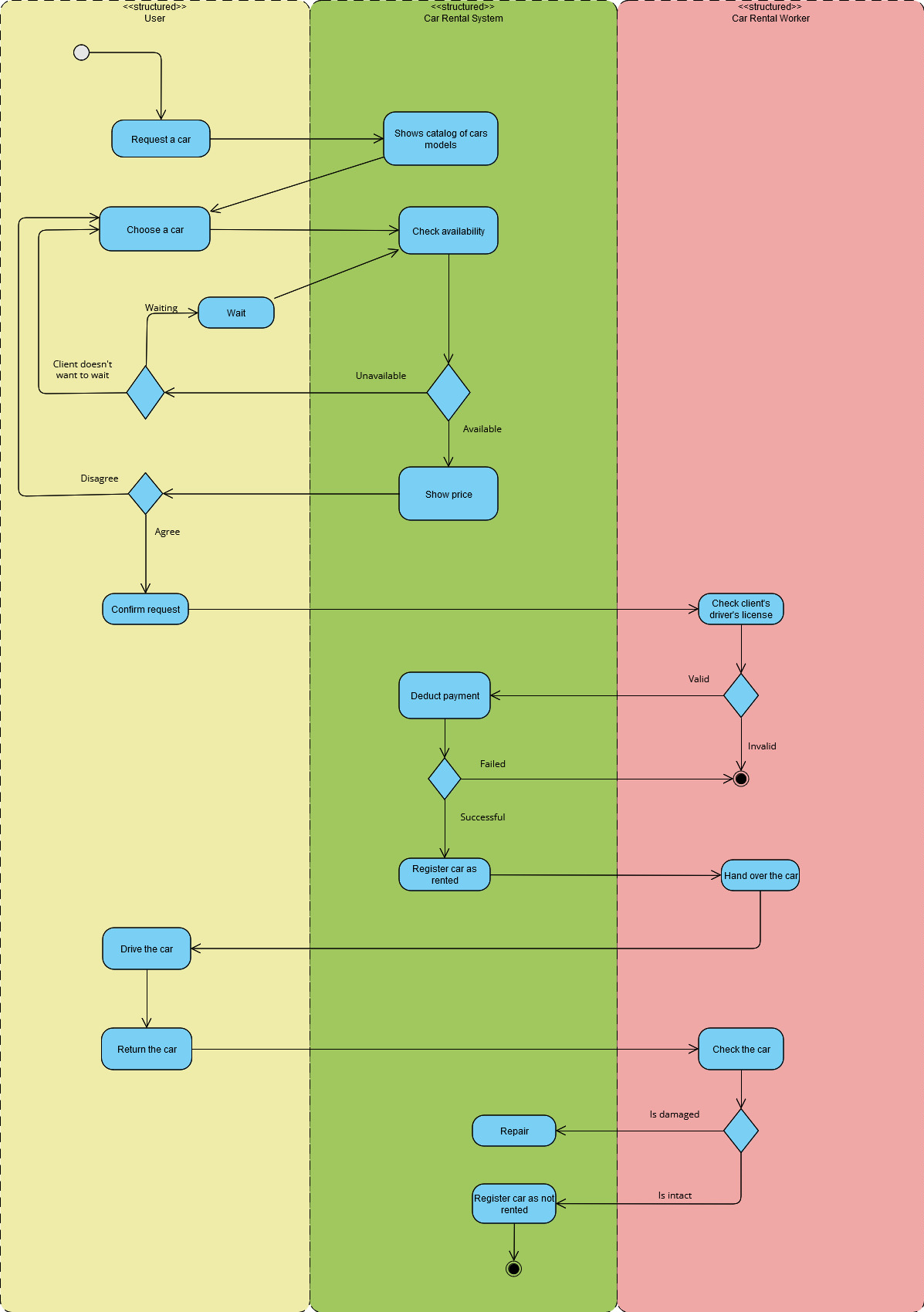
* Ima pristup funkcijama poput upravljanja automobilima i pregleda izveštaja.

#### **Korisnik (User)**

* Pretraga automobila sa opcijama za filtriranje (Search Car, Filter).
* Odabir automobila (ChooseACar).
* Plaćanje i korišćenje dodatnih usluga (Payment, Services).
* Korisnik je anoniman.

Dijagram jasno pokazuje veze između aktera i funkcionalnosti koje obavljaju, čime pruža pregled osnovnih interakcija unutar sistema.

##### **Dijagram Aktivnosti**



Ovaj dijagram aktivnosti prikazuje proces iznajmljivanja automobila između korisnika, sistema za iznajmljivanje automobila i radnika.

1. **Korisnik**:
   * Podnosi zahtev za automobil.
   * Pregleda dostupne modele i bira automobil.
   * Ako je automobil dostupan, korisnik dobija cenu i potvrđuje zahtev.
2. **Sistem za iznajmljivanje automobila**:
   * Prikazuje katalog automobila.
   * Proverava dostupnost automobila.
   * Obrađuje plaćanje i registruje automobil kao iznajmljen.
3. **Radnik**:
   * Proverava vozačku dozvolu korisnika.
   * Predaje automobil korisniku.
   * Po povratku automobila, proverava da li je automobil oštećen.
4. **Ishodi**:
   * Ako je automobil oštećen, ide na popravku.
   * Ako je u dobrom stanju, registruje se kao vraćen.

Dijagram jasno definiše tok rada i interakciju između korisnika, sistema i radnika.

1. **Analiza softvera**

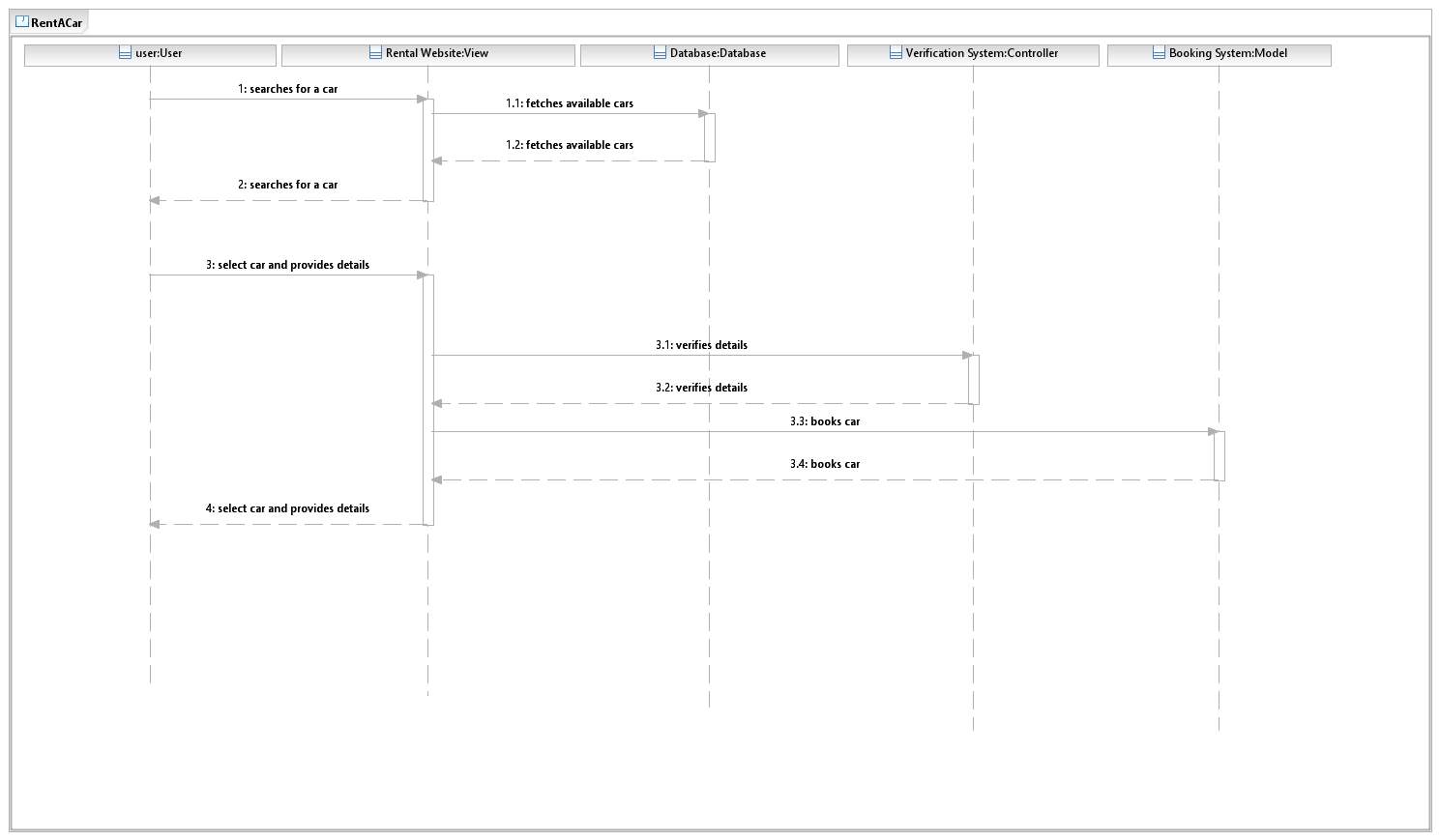
Za ovaj projekat izabrana je MVC (Model-View-Control) zbog njene modularnosti, skalabilnosti i lakog održavanja

* **Model** predstavlja podatke i logiku poslovanja, kao što su informacije o automobilima, korisnicima, rezervacijama i platnim transakcijama. Svi podaci su centralizovani i lako dostupni za obradu.
* **View** je zadužen za interakciju sa korisnicima, omogućavajući jednostavno pretraživanje i rezervaciju vozila putem korisničkog interfejsa, koji može biti web ili mobilna aplikacija.
* **Controller** upravlja komunikacijom između Modela i View-a, obrađujući korisničke zahteve (npr. rezervacija automobila) i ažurirajući interfejs na osnovu rezultata.

Ova arhitektura omogućava jasnu separaciju odgovornosti, što olakšava testiranje i dalji razvoj, kao što su dodavanje novih funkcionalnosti (poput integracije sa GPS servisima ili naprednog filtriranja vozila).



##### **Dijagram Sekvenci**



Ovaj dijagram sekvence prikazuje proces rezervacije automobila putem sistema "RentACar". Sekvencijalni dijagram koristi se za predstavljanje toka komunikacije između različitih aktera i sistema. Evo objašnjenja koraka u dijagramu:

1. **Pretraga automobila od strane korisnika:**

* **(1)** Korisnik započinje proces tako što pretražuje dostupne automobile.
* **(1.1)** Sistem šalje zahtev ka bazi podataka za dohvat informacija o dostupnim automobilima.
* **(1.2)** Baza podataka vraća informacije o dostupnim automobilima nazad ka sistemu, koji ih prikazuje korisniku.

2. **Ponovljena pretraga (ako je potrebna):**

* **(2)** Korisnik može ponovo pretraživati automobile, a proces se ponavlja kao i u prvom koraku.

3. **Izbor automobila i unos detalja:**

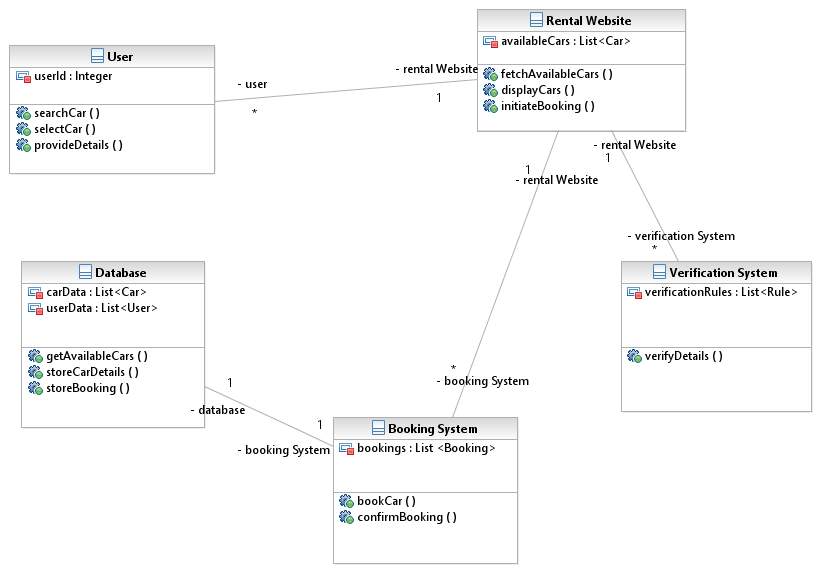
* **(3)** Korisnik bira određeni automobil i unosi tražene detalje (poput datuma rezervacije, ličnih podataka itd.).
* **(3.1)** Sistem šalje ove detalje verifikacionom modulu.
* **(3.2)** Verifikacioni modul proverava ispravnost podataka i vraća potvrdu nazad ka sistemu.

4. **Rezervacija automobila:**

* **(3.3)** Nakon verifikacije, sistem upućuje zahtev za rezervaciju automobila ka modelu rezervacija.
* **(3.4)** Rezervacioni model potvrđuje rezervaciju i šalje informaciju nazad ka korisniku.

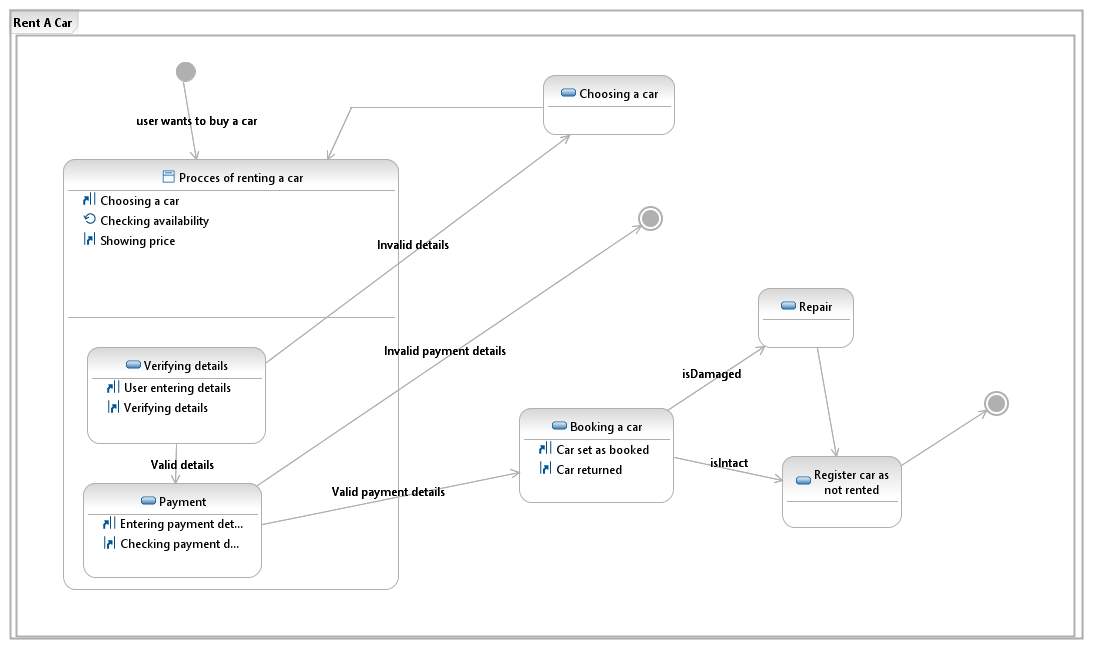
Korisnik pretražuje, bira automobil, unosi podatke, a sistem verifikuje informacije i završava proces rezervacijom.

**4.2** Logički Dijagram Klase



# Projektovanje Softvera

**5.1** Dijagram Stanja



Ovaj dijagram stanja pokazuje celokupni proces iznajmljivanja automobile.

### 1. ****Početno stanje****

* Proces započinje kada korisnik izrazi želju da iznajmi automobil (user wants to buy a car).
* Ovo stanje vodi ka glavnom procesu iznajmljivanja automobila.

### 2. ****Proces iznajmljivanja automobila****

Ovaj proces se sastoji od nekoliko međusobno povezanih koraka:

#### **a) Biranje automobila (Choosing a car)**

* Korisnik bira automobil, proverava se dostupnost vozila (Checking availability), i vidi cenu (Showing price).
* Ako su podaci korisnika neispravni (Invalid details), proces se prekida i vraća na unos detalja.

#### b) **Verifikacija detalja** (Verifying details)

* Korisnik unosi svoje lične podatke (User entering details).
* Ti podaci se proveravaju (Verifying details).
* Ako su detalji nevalidni (Invalid details), proces se vraća na unos detalja.
* Ako su detalji validni (Valid details), prelazi se na plaćanje.

#### c) **Plaćanje** (Payment)

* Korisnik unosi podatke za plaćanje (Entering payment details).
* Sistem proverava validnost tih podataka (Checking payment details).
* Ako je plaćanje nevažeće (Invalid payment details), korisnik mora ponovo uneti podatke.
* Ako je plaćanje uspešno (Valid payment details), automobil se rezerviše.

### 3. ****Rezervacija automobila**** (Booking a car)

* Sistem označava automobil kao rezervisan (Car set as booked).
* Nakon završetka korišćenja, automobil se vraća (Car returned).
* Nakon povratka automobila, postoje dve opcije:
  + Ako je automobil oštećen (isDamaged), prelazi se na proces popravke (Repair).
  + Ako je automobil neoštećen (isIntact), registruje se kao dostupan za iznajmljivanje (Register car as not rented).

### 4. ****Popravka automobila**** (Repair)

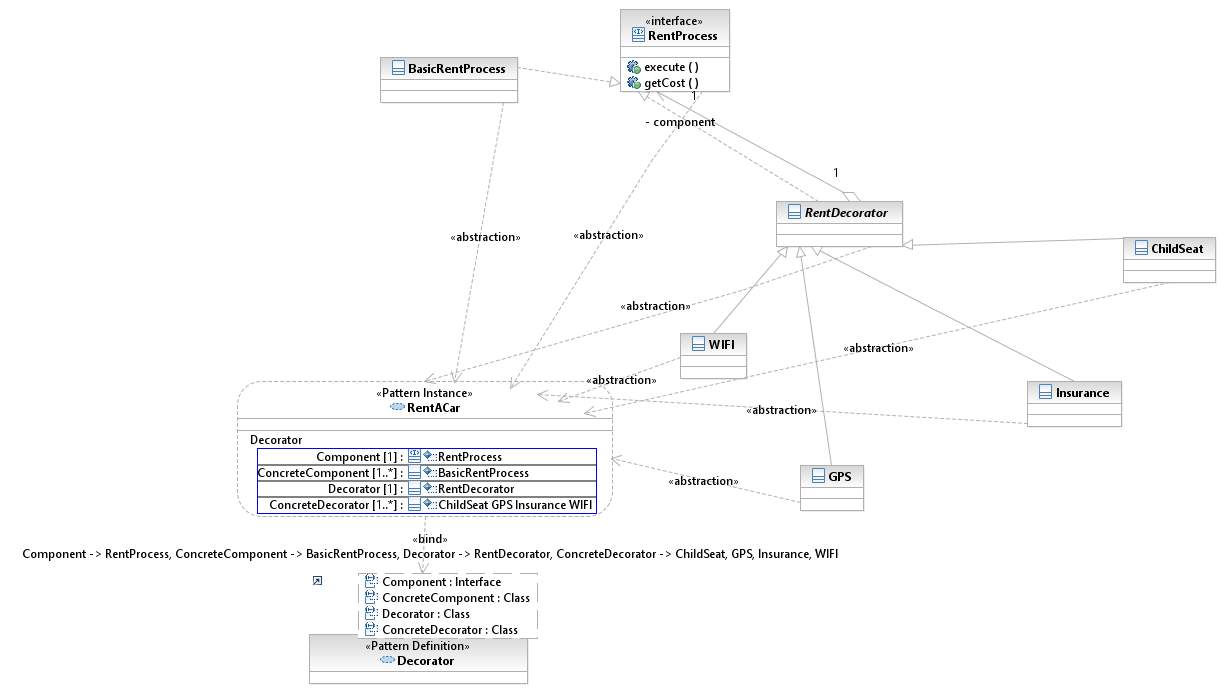
* Automobil prolazi kroz proces popravke.
* Nakon popravke, ponovo se registruje kao dostupan za iznajmljivanje (Register car as not rented).

### 5. ****Krajnje stanje****

Proces završava kada je automobil registrovan kao dostupan za iznajmljivanje ili nakon popravke.

Ovaj dijagram jasno prikazuje tok različitih stanja kroz koje proces prolazi, uz razne grane u zavisnosti od ispravnosti unetih podataka i stanja automobila.

**5.2** Dizajn Patern



Ovaj dijagram predstavlja **UML klasni dijagram** koji prikazuje primenu **Decorator dizajn šablona** na proces iznajmljivanja automobila.

1. **RentProcess (Komponenta)**:

* Ovo je **interfejs** koji definiše zajednički ugovor za sve procese iznajmljivanja automobila.
* Sadrži dve metode:
  + execute() – Izvršava proces iznajmljivanja.
  + getCost() – Vraća cenu iznajmljivanja.

2. **BasicRentProcess (Konkretna komponenta)**:

* Implementira interfejs RentProcess.
* Predstavlja osnovni proces iznajmljivanja automobila bez dodatnih opcija.
* Pruža osnovne implementacije za execute() i getCost().

3. **RentDecorator (Apstraktni dekorator)**:

* Takođe implementira interfejs RentProcess.
* Sadrži referencu na objekat tipa RentProcess (komponenta koju dekorator ukrašava).
* Služi kao osnovna klasa za sve konkretne dekoratore (npr. GPS, Insurance).

4. **Konkretni dekoratori**:

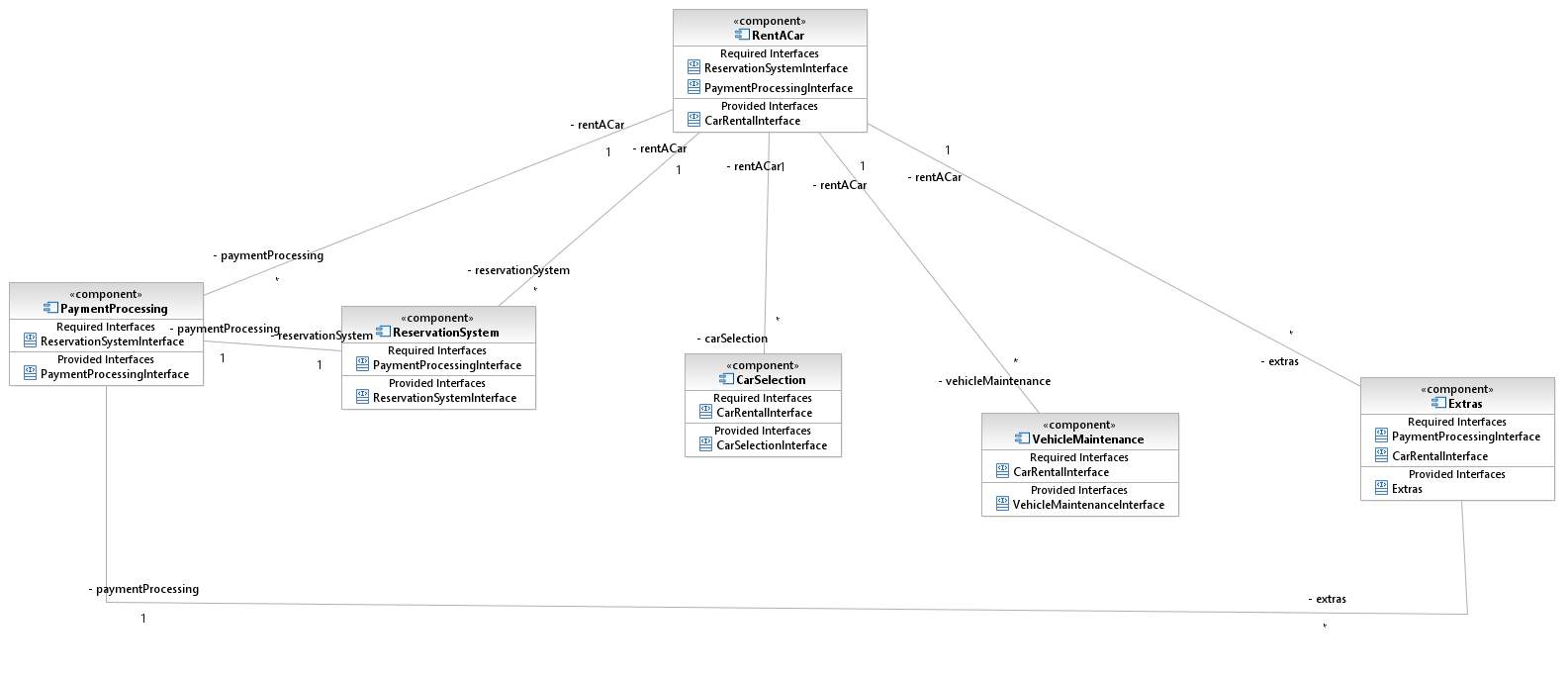
* Nasleđuju klasu RentDecorator kako bi dodali specifične funkcionalnosti.
* Primeri:
  + **GPS**: Dodaje GPS opciju u proces iznajmljivanja.
  + **ChildSeat**: Dodaje dečije sedište.
  + **Insurance**: Dodaje opciju osiguranja.
  + **WiFi**: Dodaje WiFi uslugu.
* Svaki dekorator može da nadjača metode execute() i getCost() kako bi dodao ili promenio ponašanje.

5. **Instanca šablona (RentACar)**:

* Predstavlja primenu šablona, gde su konkretne komponente i dekoratori instancirani.

Decorator šablon omogućava dinamičko dodavanje ili modifikaciju funkcionalnosti osnovnog procesa iznajmljivanja automobila bez menjanja postojećeg koda, pružajući fleksibilnost i proširivost sistema.

**5.3** Dijagram Komponenti



Ovaj dijagram prikazuje **dijagram komponenti** koji modelira različite module sistema za iznajmljivanje automobila.

1. **RentACar**

* **Pruža interfejs:** CarRentalInterface
* **Zahteva interfejse:** ReservationSystemInterface, PaymentProcessingInterface
* Predstavlja glavni modul za iznajmljivanje automobila, povezujući rezervacije i plaćanja.

2. **PaymentProcessing**

* **Pruža interfejs:** PaymentProcessingInterface
* **Zahteva interfejs:** ReservationSystemInterface
* Ova komponenta se bavi procesiranjem uplata i koristi rezervacione podatke iz sistema.

3. **ReservationSystem**

* **Pruža interfejs:** ReservationSystemInterface
* **Zahteva interfejse:** PaymentProcessingInterface
* Ovo je centralna komponenta koja upravlja rezervacijama i koristi funkcionalnosti plaćanja.

4. **CarSelection**

* **Pruža interfejs:** CarSelectionInterface
* **Zahteva interfejs:** CarRentalInterface
* Ova komponenta omogućava izbor vozila za iznajmljivanje.

5. **VehicleMaintenance**

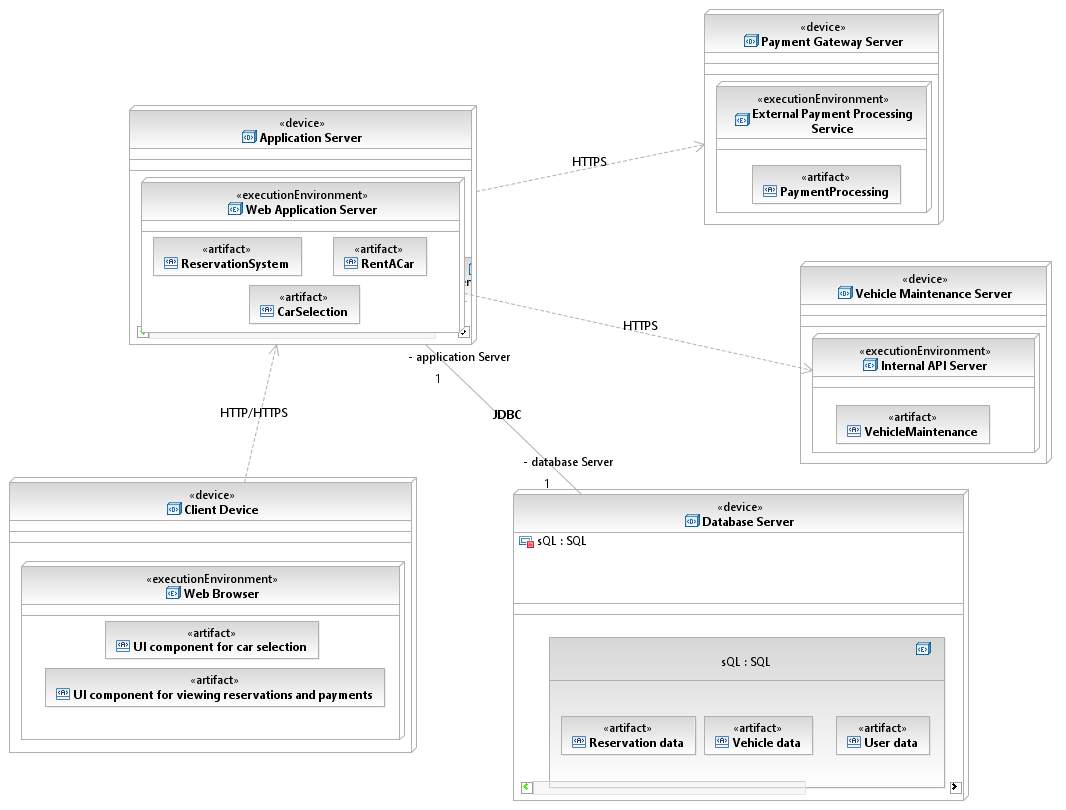
* **Pruža interfejs:** VehicleMaintenanceInterface
* **Zahteva interfejs:** CarRentalInterface
* Odgovorna je za praćenje i održavanje vozila.

6. **Extras**

* **Pruža interfejs:** Extras
* **Zahteva interfejse:** PaymentProcessingInterface, CarRentalInterface
* Koristi se za dodatne usluge (poput GPS-a, WIFI-a, sedišta za decu...).

Dijagram modelira arhitekturu sistema za iznajmljivanje automobila s jasno definisanim odgovornostima između komponenti. Fokus je na modularnosti i korišćenju interfejsa za povezivanje funkcionalnosti.

**5.4** Dijagram Uvođenja



Ovo je **UML dijagram razmeštaja (Deployment Diagram)** za sistem za iznajmljivanje automobila, koji prikazuje razmeštaj softverskih artifakta na različite uređaje i njihove okoline izvršavanja.

#### **1. Klijentski uređaj**

* **Uređaj:** Predstavlja uređaj krajnjeg korisnika (npr. računar, tablet ili pametni telefon).
* **Okruženje izvršavanja:** **Web browser** koji pokreće korisnički interfejs aplikacije.
* **Artefakti:**
  + UI komponenta za izbor automobila: Upravlja funkcionalnošću pregleda i izbora automobila.
  + UI komponenta za pregled rezervacija i plaćanja: Upravlja pregledom detalja rezervacija i plaćanja.

#### **2. Aplikacioni server**

* **Uređaj:** Predstavlja centralni server na kojem se nalazi osnovna logika aplikacije.
* **Okruženje izvršavanja:** **Web aplikacioni server** koji obrađuje poslovnu logiku i odgovara na zahteve korisnika.
* **Artifakt:**
  + ReservationSystem: Upravljanje rezervacijama, otkazivanjima i dostupnošću vozila.
  + RentACar: Koordinacija procesa iznajmljivanja automobila i integracija sa drugim servisima.
  + CarSelection: Logika za pregled i izbor vozila.

#### **3. Server za platni promet**

* **Uređaj:** Predstavlja eksterni servis za obradu plaćanja.
* **Okruženje izvršavanja:** **Eksterni servis za obradu plaćanja** (najčešće oblak, npr. Stripe ili PayPal).
* **Artifakt:**
  + PaymentProcessing: Obrada sigurnih transakcija i verifikacija plaćanja.

#### **4. Server za održavanje vozila**

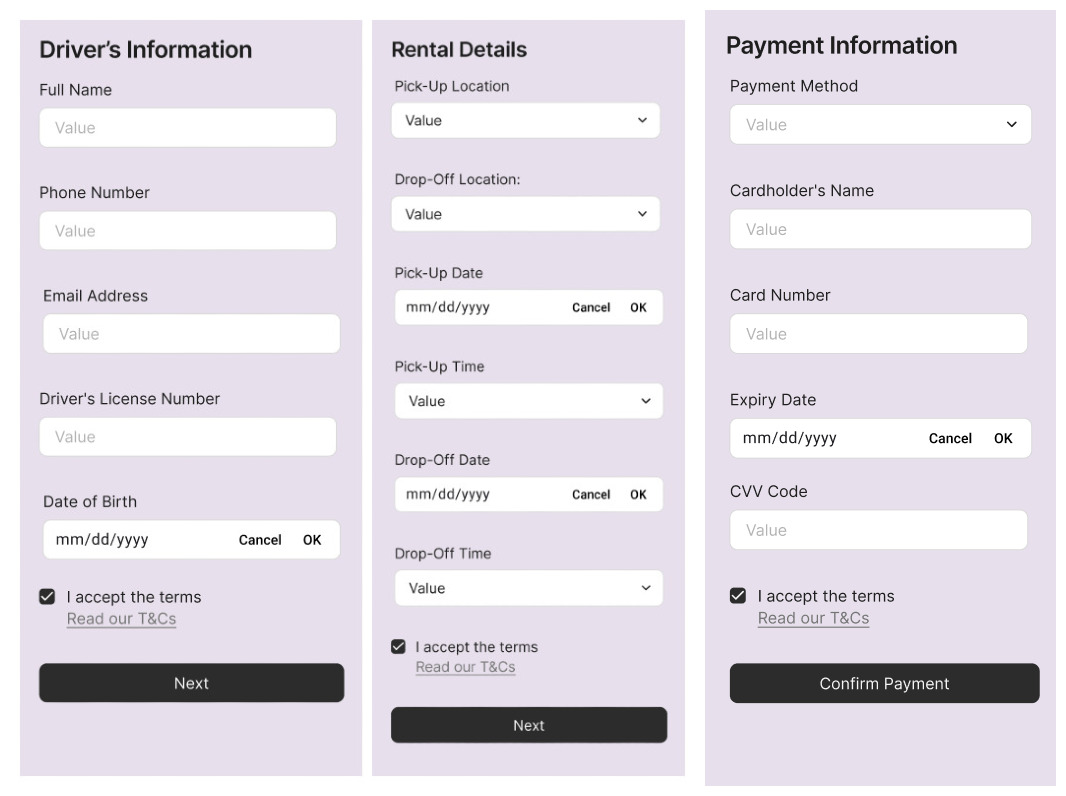
* **Uređaj:** Server zadužen za upravljanje procesima održavanja vozila.
* **Okruženje izvršavanja:** **Interni API server** za razmenu podataka o održavanju.
* **Artifakt:**
  + VehicleMaintenance: Upravljanje podacima o održavanju i servisima vozila.

#### **5. Serverska baza podataka**

* **Uređaj:** Server na kojem se nalazi baza podataka.
* **Okruženje izvršavanja:** **SQL baza podataka**
* **Artifakti:**
  + Reservation data: Podaci o rezervacijama.
  + Vehicle data: Podaci o vozilima.
  + User data: Podaci o korisnicima.

Ovaj dijagram jasno prikazuje razmeštaj softverskih komponenti, njihovu svrhu i način komunikacije između njih.

# Projektovanje slojeva prezentacije



Ova slika predstavlja prototip korisničkog interfejsa za **rent-a-car** aplikaciju i prikazuje različite delove forme podeljene u logične korake za unos informacija.

### ****1. Driver's Information (Informacije o vozaču)****

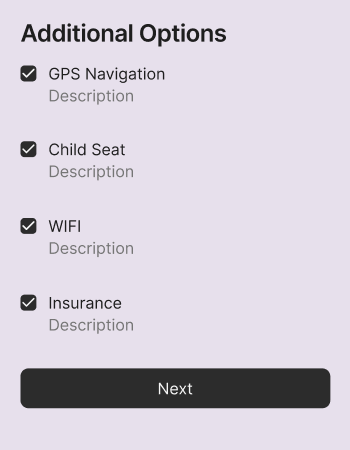
* Sadrži polja za unos osnovnih informacija o korisniku, kao što su:
  + **Full Name**: Ime i prezime.
  + **Phone Number**: Broj telefona.
  + **Email Address**: Email adresa.
  + **Driver's License Number**: Broj vozačke dozvole.
  + **Date of Birth**: Datum rođenja.
* Korisnik mora potvrditi uslove korišćenja (checkbox sa linkom za "Terms and Conditions").
* Na dnu je **Next** dugme koje vodi korisnika na sledeći korak.

### ****2. Rental Details (Detalji o iznajmljivanju)****

* U ovom koraku korisnik bira ključne informacije o iznajmljivanju vozila:
  + **Pick-Up Location**: Lokacija preuzimanja vozila.
  + **Drop-Off Location**: Lokacija vraćanja vozila.
  + **Pick-Up Date & Time**: Datum i vreme preuzimanja.
  + **Drop-Off Date & Time**: Datum i vreme vraćanja.
* Takođe postoji checkbox za prihvatanje uslova korišćenja.
* **Next** dugme vodi korisnika na sledeći korak.

### ****3. Payment Information (Informacije o plaćanju)****

* Korisnik unosi podatke za plaćanje, uključujući:
  + **Payment Method**: Način plaćanja (dropdown meni).
  + **Cardholder's Name**: Ime vlasnika kartice.
  + **Card Number**: Broj kartice.
  + **Expiry Date**: Datum isteka kartice.
  + **CVV Code**: Sigurnosni kod kartice.
* Postoji checkbox za prihvatanje uslova korišćenja, što je standardni zahtev kod transakcija.
* Dugme **Confirm Payment** završava transakciju i vodi korisnika dalje.



### ****4. Additional Options (Dodatne opcije)****

* Ovde korisnik bira dodatke za iznajmljivanje automobila pomoću **checkbox** opcija:
  + **GPS Navigation**: Navigacija putem GPS-a.
  + **Child Seat**: Dečije sedište.
  + **WIFI**: Pristup internetu putem Wi-Fi uređaja u vozilu.
  + **Insurance**: Osiguranje vozila.
* Na dnu je **Next** dugme koje vodi korisnika na sledeći korak.

### ****Analiza sloja prezentacije****

* Dizajn je **korak-po-korak proces** koji korisnika vodi kroz unos informacija, što poboljšava korisničko iskustvo.
* Koriste se jasne i standardne forme za unos podataka sa intuitivnim tekstualnim poljima i checkbox opcijama.
* Održava se konzistentan vizuelni stil sa neutralnim bojama i fokusom na funkcionalnost.

# 7. Zaključak

Razvoj aplikacije za rentiranje automobila korišćenjem metodologije *Design Thinking* pokazao se kao efikasan pristup za rešavanje ključnih izazova sa kojima se korisnici suočavaju prilikom iznajmljivanja vozila. Ova metodologija omogućila je detaljno razumevanje potreba korisnika kroz fazu empatije i iterativan proces razvoja, što je rezultiralo intuitivnim i funkcionalnim rešenjem.

Prototip aplikacije izrađen je na osnovu povratnih informacija korisnika, čime je osigurano da aplikacija odgovara njihovim zahtevima i očekivanjima. Korišćenje *MVC arhitekture* doprinelo je modularnosti i skalabilnosti sistema, omogućavajući lako održavanje i dalji razvoj. Implementacija *Decorator dizajn šablona* omogućila je fleksibilnost pri dodavanju novih funkcionalnosti, poput dodatnih usluga (GPS, osiguranje i WiFi).

Rezultati testiranja prototipa pokazali su da sistem efikasno rešava identifikovane probleme, poput složenosti pretrage i nejasnosti cena. Korisnički interfejs dizajniran je tako da pruža korak-po-korak vodič kroz proces rezervacije, čime se poboljšava iskustvo korisnika.

Ovaj projekat pokazuje kako kombinacija savremenih metodologija, kao što je *Design Thinking*, i inovativnih softverskih rešenja može unaprediti procese u industriji rentiranja automobila, stvarajući aplikaciju koja je intuitivna, brza i pouzdana. Dalji razvoj može uključivati napredne funkcionalnosti, poput integracije sa GPS sistemima, analitike korisničkog ponašanja i personalizovanih preporuka, što bi dodatno poboljšalo iskustvo korisnika i povećalo konkurentnost aplikacije na tržištu.

# 8. Literatura

1. [Mockflow](https://mockflow.com/) [Wireframing](https://mockflow.com/)
2. [Ict Hub – Design Thinking metodologija](https://www.icthub.rs/clanak/design-thinking-metodologija-pet-glavnih-faza-u-radu)
3. [Royal – Rent a Car Kraljevo](https://rentacarkraljevo77.rs/)
4. [Location Generator](https://www.fakepersongenerator.com/random-address)
5. [Program Strategy HQ](https://www.programstrategyhq.com/post/design-thinking-process)
6. [Software Ideas Modeler](https://www.softwareideas.net/)
7. [Visual Paradigm Online](https://online.visual-paradigm.com/drive/#proj=0&diagram=list)
8. [Mmdn web docs - MVC](https://developer.mozilla.org/en-US/docs/Glossary/MVC)
9. [Figma.com - UI](https://www.figma.com)

[10] [Pine Tools – Merge Images](https://pinetools.com/merge-images)