QUICK

SERVICE AT HOME

Field Service Application

CS201/261 Project by,

Renish R
(202252333)
Utkarsh Rana
(202251148)
Vedulla SaiVardhan
(202251156)

Yallapu Vignesh
(202251161)
Vankayala Koutilya
(202251152)
Kompalli Akhil
(202252324)

Professor: Dr.Novarun Deb

Introduction

Quack Origin

The name ”Quack” originates from our commitment to QUick ACKcess in field ser- vices. It embodies speed and accessibility, reflecting our dedication to prompt solutions. The playful name adds a unique and memorable touch, distinguishing our application in the market.

Objectives and Goals

The primary objectives of the Quack field service application are to revolutionize and streamline the delivery of field services by providing swift and efficient access to technicians for customers. We aim to enhance the overall customer experience by reducing response times and ensuring a seamless communication channel between technicians and clients. Quack strives to be a user-friendly and reliable platform that not only meets but exceeds the expectations of both service providers and customers.

Our goals include improving operational efficiency, increasing customer satisfaction, and establishing Quack as a leading solution in the field service industry. Through continuous innovation and a customer-centric approach, we aspire to contribute to the success and growth of businesses relying on field service operations.

Key Features of Our WebSite:

- Transparent Pricing: The app prioritizes transparency by providing users with clear and easily understandable pricing information.
- Real-Time Bidding: Implements a real-time bidding system, enhancing user convenience and facilitating efficient service provider selection.
- User Reviews and Ratings: Trust-building is emphasized through a robust system of user reviews and ratings for service providers.
- Convenient Platform: A user-friendly platform streamlines the service request process, making it easy for users to post problems, receive bids, and communicate with workers.

Project Overview

- This project revolves around the development of a web application designed for hiring technicians to address home-based problems and repairs.
- There are two distinct user roles within the website:

  1. Clients: Users who seek technicians for their specific requirements.
  2. Technicians: Service providers who offer their expertise to clients.

- Both clients and technicians are required to register accounts, provide proof of identity, and specify their locations.
- The application caters to various technician specialties, including:

  – Electricians

  – Mechanics

  – Plumbers

  – Carpenters, etc.

- Clients can submit service requests by choosing from predefined options and providing additional details such as pictures, videos, and descriptions.
- Once a service request is submitted, designated technicians in the same locality receive the request and can review the problem.
- Technicians have the ability to set service charges for the request and send the details to the client.
- Clients can review the service charges from various technicians and accept an offer from the preferred service provider.
- Clients can also view ratings and reviews of technicians to aid in their decision-making process.
- After choosing a technician, clients receive contact details for communication.
- Upon successful completion of the service, clients can make payments to technicians
- through an integrated payment option within the app.
- The admin of the platform receives a small portion of the payment as a platform charge.

User Statistics

Type of End-Users

Determining the number of end users for an app that is not yet fully developed can be challenging.As our app is user friendly and simple to use. Our main target audience are

Old-age people as it is user-friendly.

- People living alone in a house as they might have much work in their daily-
- lives.
- People who moved to a new place as language or knowing the people might be lacking.

Even the technicians are selected based on their rating and money demanded, So customers will get service in a reasonable cost. The cost might be increased for overtime work of technician which will be paid by customer. As our app have many features it makes it simple and user-friendly. We can expect around 50k+ including technicians. As this is a rough estimation the users might be more than expected if the app gets updated to the market needs and might also be less if not maintained well.

Age group Analysis

Determining the age group of end users for the application would depend on various factors, including the nature of the services provided and the industry targeted. However, if we consider a broad and general estimate, we might expect a diverse range of end users spanning from late teens to middle-aged adults.

• Young Adults (18-24): These users might include tech-savvy individuals seeking quick and efficient solutions for their service needs.

• Adults (25-44): This age group often has diverse service requirements, from household maintenance to professional services, making them potential users of a versatile field service application.

• Middle-Aged Adults (45-64): Individuals in this age group might be homeowners or business owners who require efficient field services for both personal and professional purposes.

Stakeholders

1. Clients (Users seeking services):

   - Description: Clients are the primary users seeking services like electricians, plumbers, carpenters, and mechanics through the app.
   - Role: Clients post service requests, receive bids, review technicians, and communicate with them. Trust-building features, such as user reviews, are crucial for their decision-making.

2. Technicians (Service Providers):

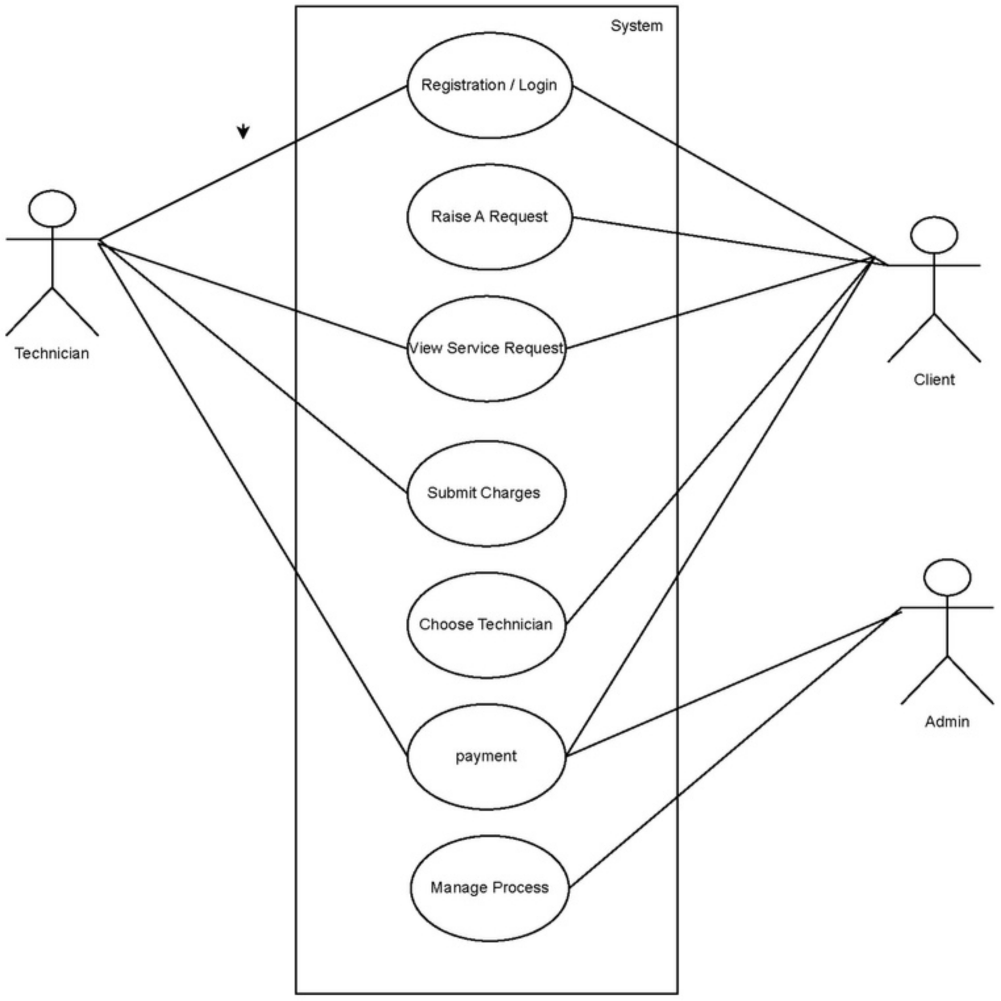
   - Description: Technicians include electricians, plumbers, carpenters, and mechanics offering services through the app.
   - Role: Technicians respond to service requests, submit bids, and communicate with clients. The bidding system fosters a competitive pricing environment, optimizing service offerings.

3. Admin (Platform Administrators):

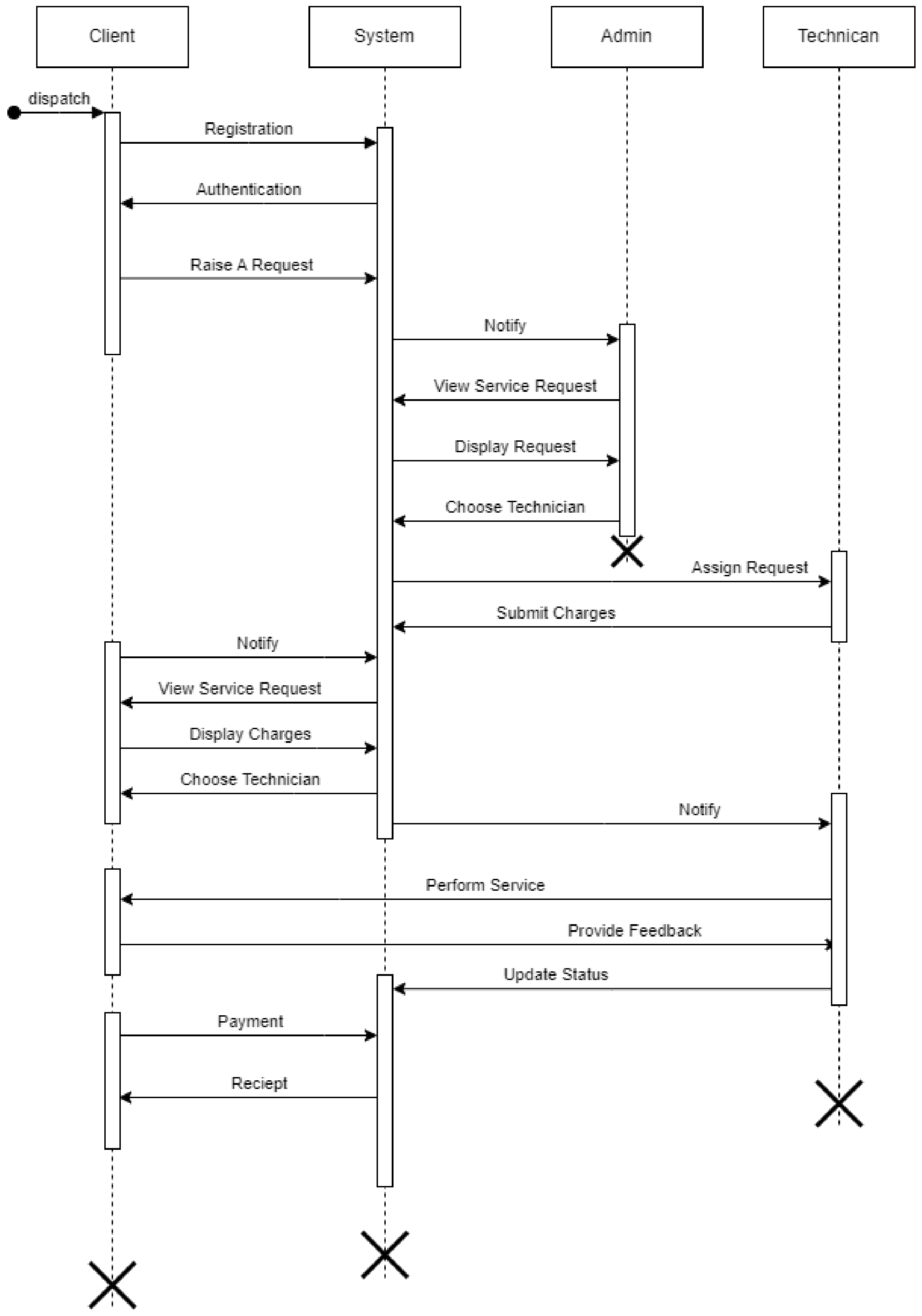
   - Description: Admins oversee the platform's functionality, managing user accounts, resolving disputes, and ensuring policy compliance.
   - Role: Admins monitor and moderate the platform, ensuring a smooth user experience. They play a crucial role in maintaining transparency, resolving issues, and upholding platform credibility.

System Design (UML)

Use Case Diagram



Sequence Diagram



Explanation

The service request process, as illustrated in the sequence diagram, unfolds through a series of well-defined steps involving key actors such as the client, system, admin, and technician. It commences with the client initiating the process by dispatching a service request to the system. Upon receipt, the system meticulously registers the request and authenticates the client to ensure a secure interaction.

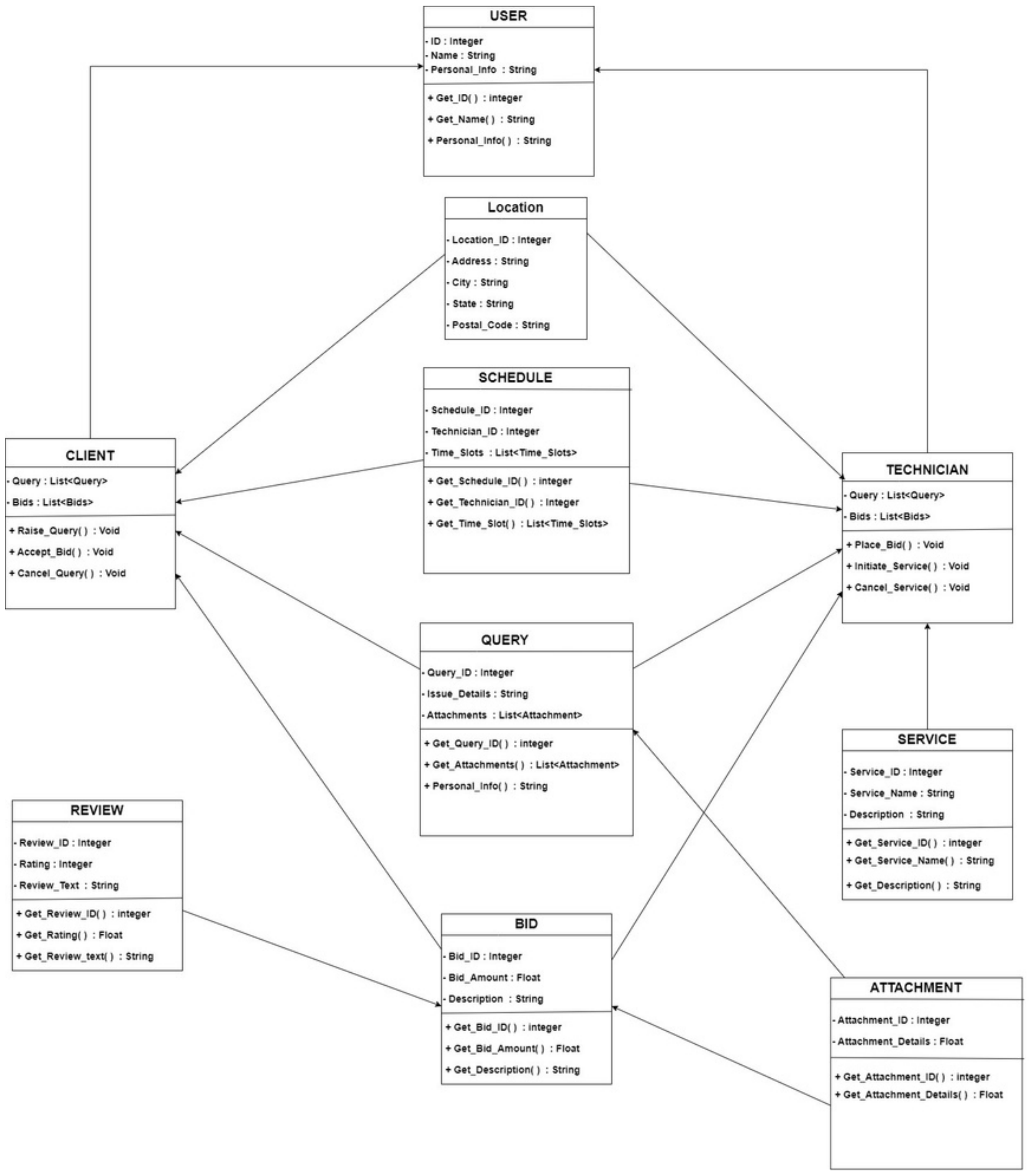
The admin, playing a crucial role in the orchestration, receives notification of the new request and gains visibility into its details. Subsequently, the admin selects a technician based on considerations of availability and expertise. Once chosen, the technician is promptly notified and receives access to the request details.

The technician, having reviewed the service request, submits the associated charges. Simultaneously, the client is notified of the assigned technician and presented with the service charges. At this juncture, the client has the option to either accept the charges and proceed with the service or decline.

Should the client opt to proceed, the technician performs the service with diligence and expertise, subsequently providing feedback on its completion. The admin, in re- sponse to the technician's feedback, updates the request status and oversees the financial aspect, receiving payment from the client.

The service request process concludes with the client receiving a receipt for the completed service, signifying the successful orchestration of the entire workflow. This orchestrated sequence ensures a streamlined and efficient experience for all parties involved, from the initial client request to the finalization of the service and financial transactions.

Class Diagram


USER
- ID : Integer
- Name : String
- Personal_Info : String
+ Get_ID( ) : integer
+ Get_Name( ) : String
+ Personal_Info( ) : String
Location
- Location_ID : Integer
- Address : String
- City : String
- State : String
- Postal_Code : String
SCHEDULE
- Schedule_ID : Integer
- Technician_ID : Integer
- Time_Slots : List<Time_Slots>
+ Get_Schedule_ID( ) : integer
+ Get_Technician_ID( ) : Integer
+ Get_Time_Slot( ) : List<Time_Slots>
CLIENT
- Query : List<Query>
- Bids : List<Bids>
+ Raise_Query( ) : Void
+ Accept_Bid( ) : Void
+ Cancel_Query( ) : Void
TECHNICIAN
- Query : List<Query>
- Bids : List<Bids>
+ Place_Bid( ) : Void
+ Initiate_Service( ) : Void
+ Cancel_Service( ) : Void
QUERY
- Query_ID : Integer
- Issue_Details : String
- Attachments : List<Attachment>
+ Get_Query_ID( ) : integer
+ Get_Attachments( ) : List<Attachment>
+ Personal_Info( ) : String
SERVICE
- Service_ID : Integer
- Service_Name : String
- Description : String
+ Get_Service_ID( ) : integer
+ Get_Service_Name( ) : String
+ Get_Description( ) : String
REVIEW
- Review_ID : Integer
- Rating : Integer
- Review_Text : String
+ Get_Review_ID( ) : integer
+ Get_Rating( ) : Float
+ Get_Review_text( ) : String
BID
- Bid_ID : Integer
- Bid_Amount : Float
- Description : String
+ Get_Bid_ID( ) : integer
+ Get_Bid_Amount( ) : Float
+ Get_Description( ) : String
ATTACHMENT
- Attachment_ID : Integer
- Attachment_Details : Float
+ Get_Attachment_ID( ) : integer
+ Get_Attachment_Details( ) : Float

Explanation

1. Client's Journey:

The client's service request journey commences with the submission of essential details, informing the system about their needs. After reviewing and confirming the request, selecting a preferred technician, and submitting service charges, the client finalizes the process by making a payment. A transaction receipt is then provided upon successful completion.

2. System's Orchestration:

The system's orchestration in the service request process involves a systematic series of steps. It begins with promptly registering the client's service request notification for official documentation. Seeking admin authorization follows, ensuring a secure workflow. Upon approval, the system verifies the request, forwards details to the chosen technician, manages client-technician communication, and handles financial aspects. This includes receiving charges, initiating admin review, and serving as a central hub for financial transactions and administrative approvals. Post-service completion, the system actively gathers feedback, updates service status, and promptly informs the client of any developments.

3. Admin's Oversight:

The system's service request orchestration follows a structured process. It starts with receiving and registering the client's request for official documentation. Admin authorization is sought to ensure a secure workflow. After approval, the system verifies the request, notifies the chosen technician, manages client-technician communication, and oversees financial transactions. Post-service completion, the system collects feedback, updates service status, and informs the client of any changes, acting as a central hub for coordination.

4. Technician's Responsibility:

The technician's role in the service request process is pivotal. They receive detailed service requests from the system and execute tasks with precision, ensuring client needs are met. After completing the service, the technician engages in a feedback loop, providing valuable insights for ongoing improvements. Sharing observations contributes to refining the entire service workflow, fostering collaboration and responsiveness. This approach prioritizes client satisfaction, emphasizing continuous enhancement in the service request process.

Potential Competitors and Challenges

Competitors in the market

1. TaskRabbit:
   - Limited Geographic Coverage: TaskRabbit may not be available in all locations, limiting access to its services for users in certain areas.
   - Skill Specialization: The platform might lack specialized professionals for certain tasks, making it challenging for users to find experts in niche areas.
   - Quality Control: Since TaskRabbit relies on individual service providers, maintaining consistent service quality across different tasks and providers may be challenging.

2. Thumbtack:
   - ServiceProviderVetting: The platform's vetting process for service providers may not be comprehensive, leading to variations in the quality of services.
   - Communication Challenges: Thumbtack's communication tools may not be robust, potentially leading to misunderstandings between clients and service providers.
   - Limited Service Customization: Some users may find the platform restrictive in terms of customizing services, as options may be predefined and less flexible.

3. Urban Company:
   - Pricing Transparency: Users might find it challenging to understand the pricing structure for various services on Urban Company, leading to potential surprises in costs.
   - Service Availability: Certain services may not be readily available at all times, especially during peak hours or in specific locations.
   - Dependence on Ratings: The reliance on user ratings may not always accurately reflect service quality, as some users might provide biased or inaccurate feedback.

Solutions to Competitors' Problems

- Our application broadens the scope by offering a platform for a diverse range of services, addressing the need for a comprehensive solution beyond specific tasks.
- Our application prioritizes user flexibility by empowering clients to have more control over service timings, enhancing the overall convenience of the hiring process.
- Our application distinguishes itself by providing a flexible scheduling system, allowing clients to have more control over service timings, thus addressing scheduling concerns.
- Our application introduces a modern and user-friendly mobile platform, streamlining the service request process. It enhances efficiency with features like real-time bidding, providing a more convenient alternative to traditional methods.

Challenges faced during Development

- Limited Geographic Coverage: Expanding the app's reach to ensure it caters to a wider audience, addressing potential limitations in service areas.
- Real-Time Bidding Implementation: Overcoming the complexity associated with developing and integrating a real-time bidding system into the app.
- Ensuring Transparency in Pricing: Designing and implementing a system that guarantees clear and transparent pricing information, promoting user trust.
- Building Trust: Proactively addressing trust issues by establishing and maintaining a reliable review and rating system for service providers.
- Mobile App Development Challenges: Overcoming typical challenges related to app development, including bug resolution, optimizing usability, and ensuring compatibility across different platforms.

Future Enhancements

In-app messaging facilitates direct user-technician communication, enabling pre-service discussions, real-time updates, and post-service support. It aids service confirmation, issue clarification, and documentation, enhancing the overall user experience and providing a convenient platform for communication.

Geo-location Upgrades

Geo-location features in mobile apps offer enhanced user experiences and service efficiency. This overview highlights missed opportunities, including precise positioning, optimized routing, map displays, and location-based services, emphasizing accuracy and privacy.

1. User Location:

• The app should have determined the user's location using GPS, Wi-Fi, or cellular data, depending on the device's capabilities.

2. Technician Location:

    • Similarly, the app should have tracked the location of available technicians to match users with nearby technicians for quicker response times.

3. Routing and Navigation:

    - Geo-location integration would have allowed the app to calculate the best route for a technician to reach the user's location efficiently.
    - This would have been particularly useful in situations where quick assistance was needed.

4. Map Display:

    - You could have incorporated a map interface within the app, showing the real-time locations of both users and technicians.
    - This visual representation could have enhanced the user experience by providing a clear understanding of the service process.

5. Location-Based Services:

- The app could have used geo-location data to offer location-specific services or information.
- For example, the app could have recommended nearby stores for purchasing replacement parts or suggested relevant services based on the user's location.

6. Check-In/Check-Out Feature:

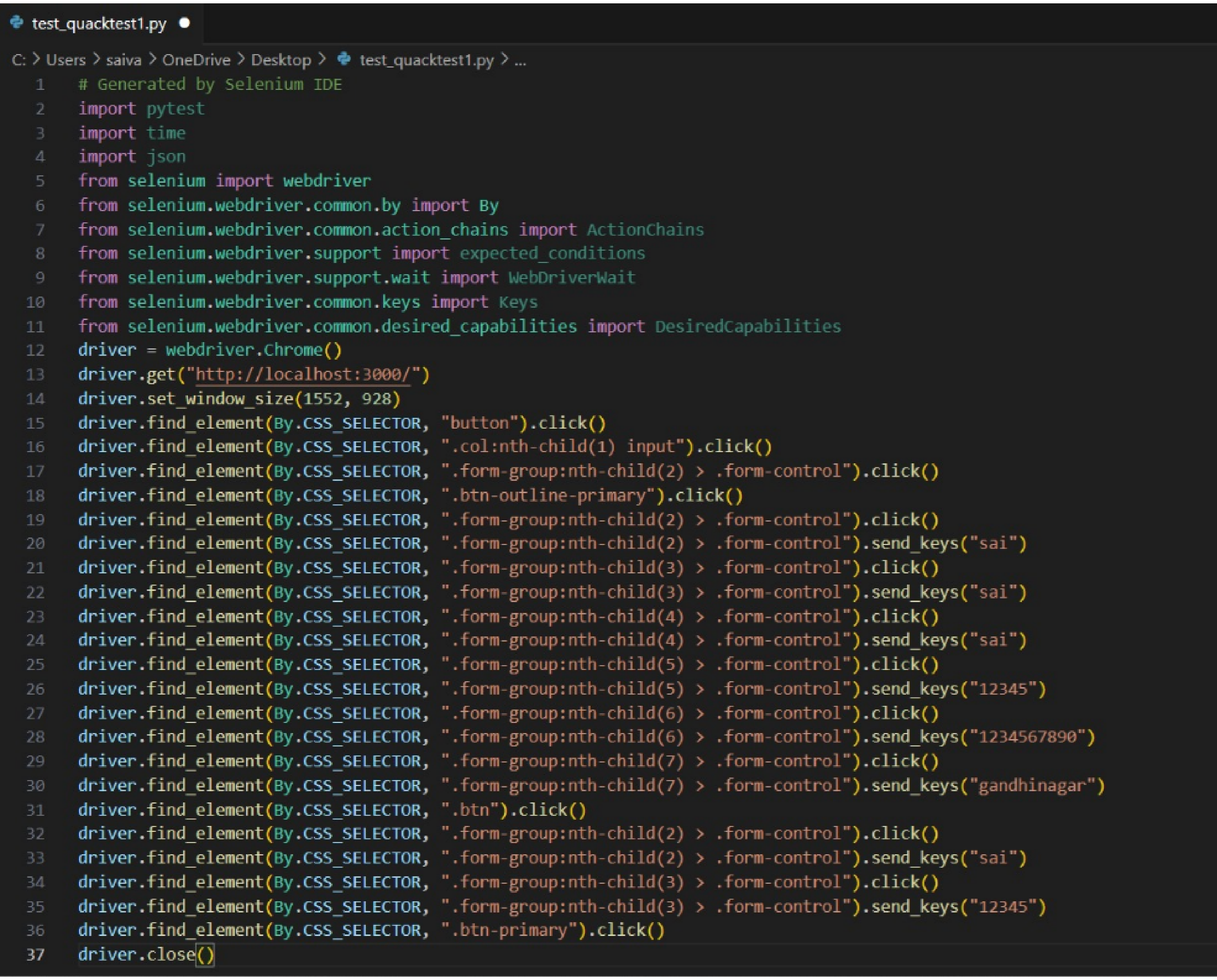
   - When a technician arrived at the user's location, they could have used the app to check in, and similarly, checked out when the service was completed.
   - This would have helped in tracking the progress of service calls.

7. Accuracy and Privacy:

   - It would have been important to ensure that the geo-location information was accurate and up-to-date.
   - Additionally, privacy would have been prioritized by providing clear information about how location data would be used and obtaining user consent.

TEST CASE 1:

```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
driver = webdriver.Chrome()
driver.get("http://localhost:3000/")
driver.set_window_size(1552, 928)
driver.find_element(By.CSS_SELECTOR, "button").click()
driver.find_element(By.CSS_SELECTOR, ".col:nth-child(1) input").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".btn-outline-primary").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-
control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-
control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(4) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(4) > .form-
control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(5) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(5) > .form-
control").send_keys("12345")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(6) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(6) > .form-
control").send_keys("1234567890")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(7) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(7) > .form-
control").send_keys("gandhinagar")
driver.find_element(By.CSS_SELECTOR, ".btn").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-
control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-
control").send_keys("12345")
driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
driver.close()
```



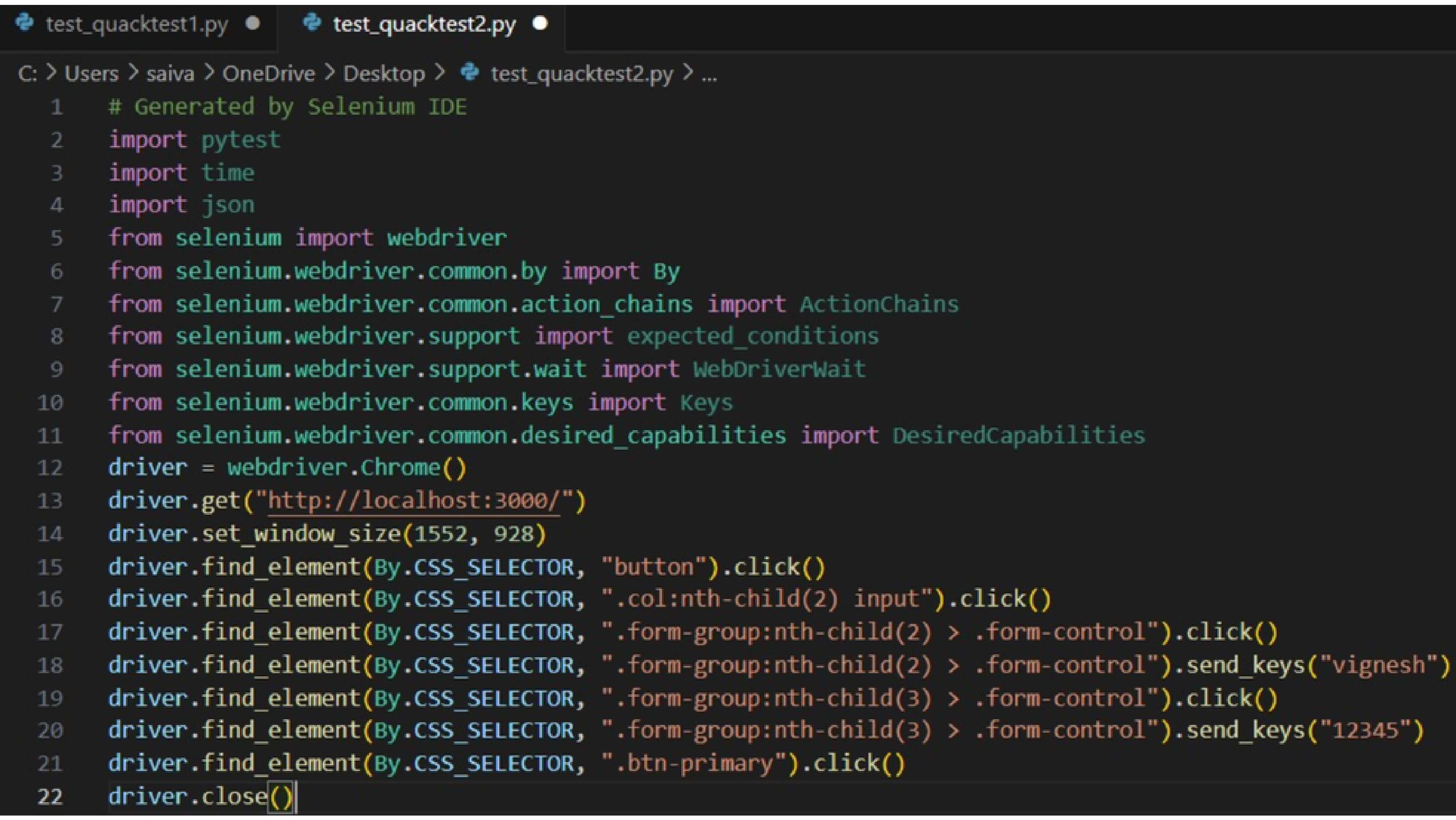
test_quacktest1.py ●

C: > Users > saiva > OneDrive > Desktop > test_quacktest1.py > ...

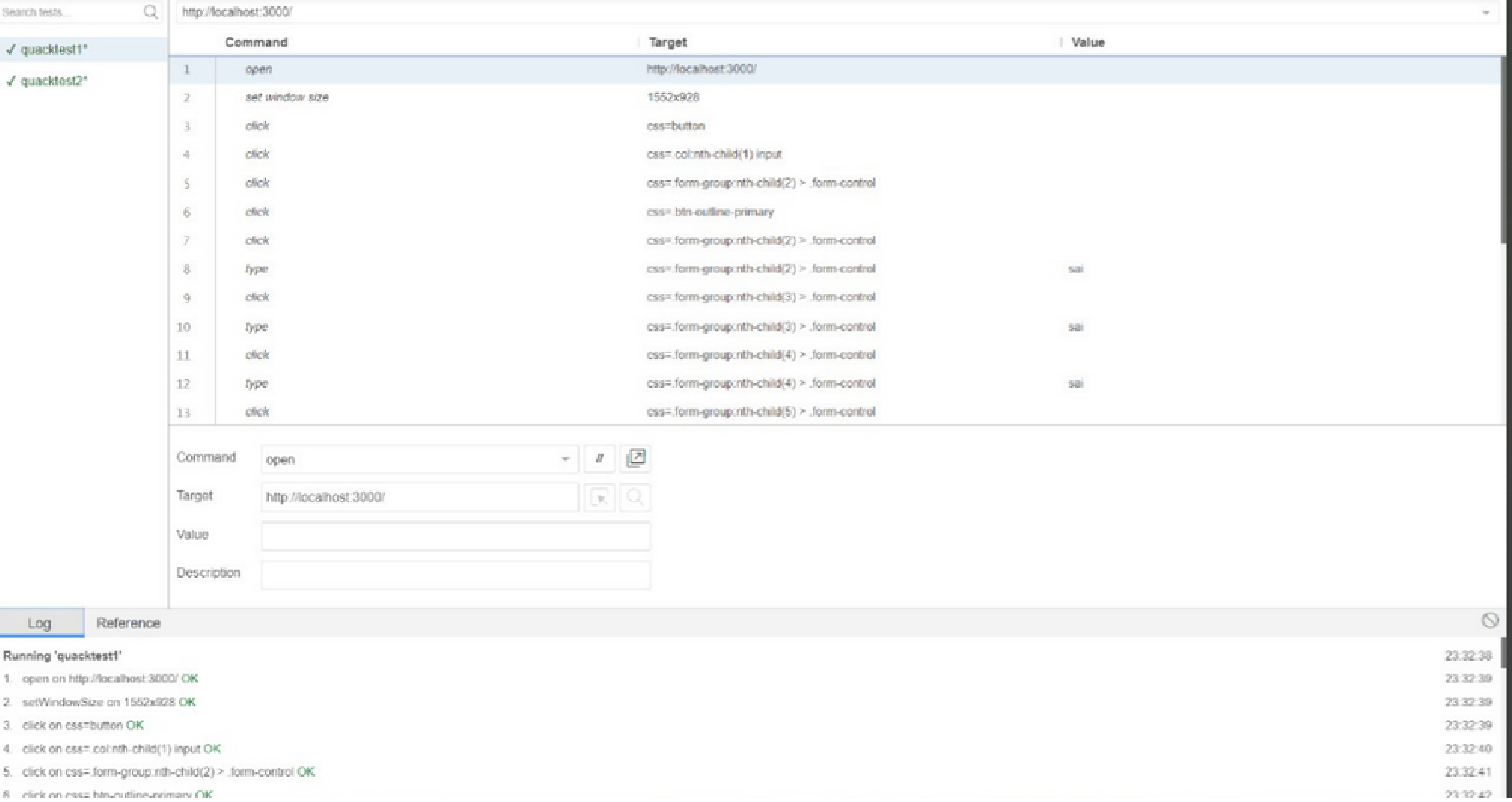
```python
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
driver = webdriver.Chrome()
driver.get("http://localhost:3000/")
driver.set_window_size(1552, 928)
driver.find_element(By.CSS_SELECTOR, "button").click()
driver.find_element(By.CSS_SELECTOR, ".col:nth-child(1) input").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".btn-outline-primary").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(4) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(4) > .form-control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(5) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(5) > .form-control").send_keys("12345")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(6) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(6) > .form-control").send_keys("1234567890")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(7) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(7) > .form-control").send_keys("gandhinagar")
driver.find_element(By.CSS_SELECTOR, ".btn").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").send_keys("sai")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").send_keys("12345")
driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
driver.close()
```

TEST CASE 2:

```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
driver = webdriver.Chrome()
driver.get("http://localhost:3000/")
driver.set_window_size(1552, 928)
driver.find_element(By.CSS_SELECTOR, "button").click()
driver.find_element(By.CSS_SELECTOR, ".col:nth-child(2) input").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-
control").send_keys("vignesh")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-
control").send_keys("12345")
driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
driver.close()
```


test_quacktest1.py
test_quacktest2.py
C: > Users > saiva > OneDrive > Desktop > test_quacktest2.py > ...

```python
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
driver = webdriver.Chrome()
driver.get("http://localhost:3000/")
driver.set_window_size(1552, 928)
driver.find_element(By.CSS_SELECTOR, "button").click()
driver.find_element(By.CSS_SELECTOR, ".col:nth-child(2) input").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(2) > .form-control").send_keys("vignesh")
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").click()
driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3) > .form-control").send_keys("12345")
driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
driver.close()
```

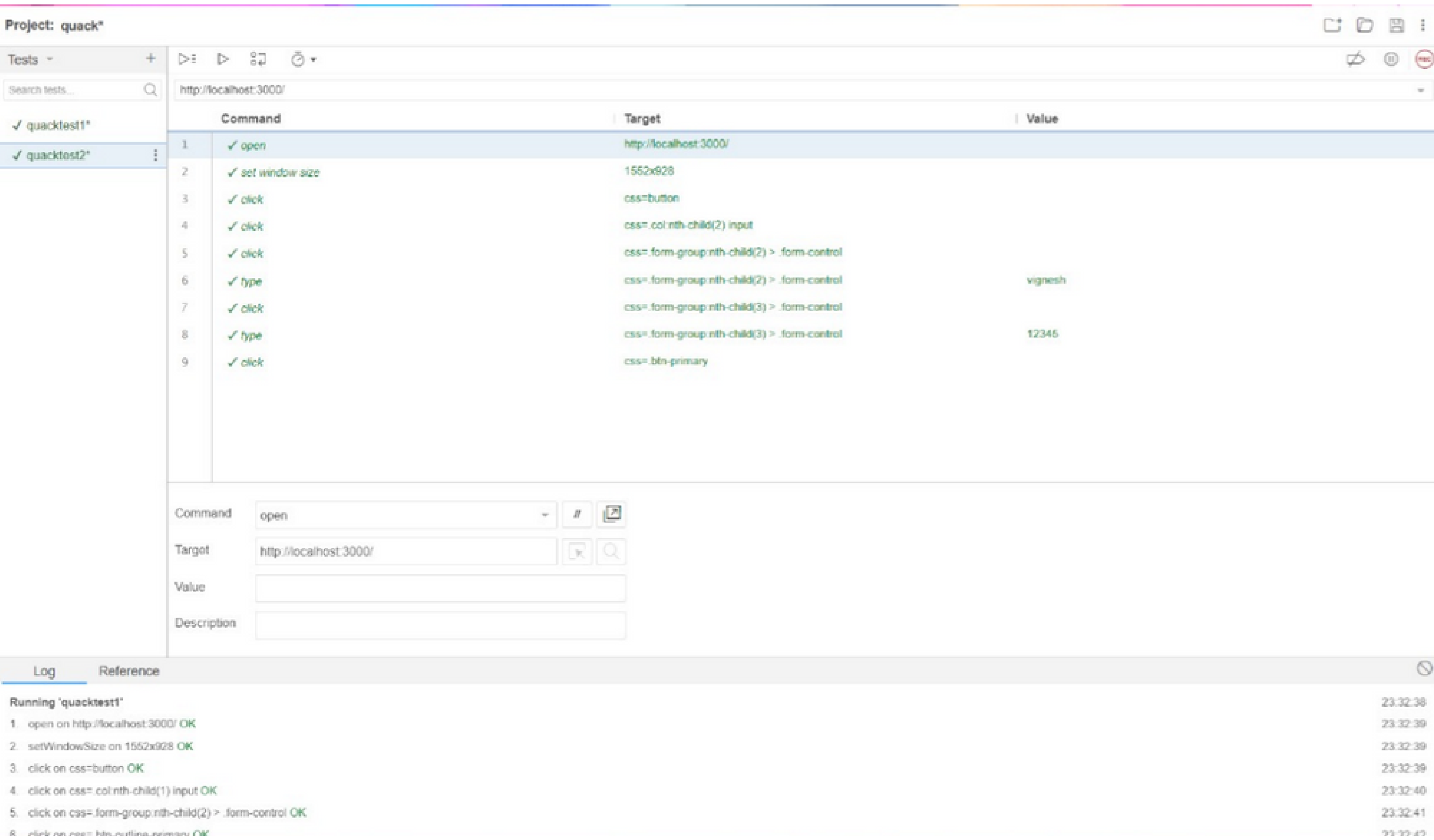


http://localhost:3000/

	Command	Target	Value
1	open	http://localhost:3000/	
2	set window size	1552x928	
3	click	css=button	
4	click	css=.col:nth-child(1) input	
5	click	css=.form-group:nth-child(2) > .form-control	
6	click	css=.btn-outline-primary	
7	click	css=.form-group:nth-child(2) > .form-control	
8	type	css=.form-group:nth-child(2) > .form-control	sai
9	click	css=.form-group:nth-child(3) > .form-control	
10	type	css=.form-group:nth-child(3) > .form-control	sai
11	click	css=.form-group:nth-child(4) > .form-control	
12	type	css=.form-group:nth-child(4) > .form-control	sai
13	click	css=.form-group:nth-child(5) > .form-control	

Command	open
Target	http://localhost:3000/
Value	
Description	

```
Running 'quacktest1'                                                    23:32:38
1. open on http://localhost:3000/ OK                                    23:32:39
2. setWindowSize on 1552x928 OK                                         23:32:39
3. click on css=button OK                                               23:32:39
4. click on css=.col:nth-child(1) input OK                              23:32:40
5. click on css=.form-group:nth-child(2) > .form-control OK             23:32:41
6. click on css=.btn-outline-primary OK                                 23:32:42
```



http://localhost:3000/

	Command	Target	Value
1	✓ open	http://localhost:3000/	
2	✓ set window size	1552x928	
3	✓ click	css=button	
4	✓ click	css=.col:nth-child(2) input	
5	✓ click	css=.form-group:nth-child(2) > .form-control	
6	✓ type	css=.form-group:nth-child(2) > .form-control	vignesh
7	✓ click	css=.form-group:nth-child(3) > .form-control	
8	✓ type	css=.form-group:nth-child(3) > .form-control	12345
9	✓ click	css=.btn-primary	

Command	open
Target	http://localhost:3000/
Value	
Description	

```
Running 'quacktest1'                                                    23:32:38
1. open on http://localhost:3000/ OK                                    23:32:39
2. setWindowSize on 1552x928 OK                                         23:32:39
3. click on css=button OK                                               23:32:39
4. click on css=.col:nth-child(1) input OK                              23:32:40
5. click on css=.form-group:nth-child(2) > .form-control OK             23:32:41
6. click on css=.btn-outline-primary OK                                 23:32:42
```