Introduction to Software Engineering Portfolio

# Descriptive Report on Software Artefacts

Users

|  |  |  |
| --- | --- | --- |
| User | Role | Persona |
| Bob | New User | Bob is a university student who has just had his car taken in for repairs, which could take up to a week. He knows he will need a taxi to get to University so he wants to create an account and enter his details in preparation for next week. |
| Derek | Existing User | Derek is a commuter and regular user of SHUBER and often uses it to travel in and around the city. Derek also relies on SHUBER to commute to work. |

User Stories

|  |  |  |
| --- | --- | --- |
| User | User Story | Acceptance Tests |
|  | Bob wants to create an account so that he can access the application. | * If a field is left empty, verify that Bob is prompted to enter in all details, and an account is not created. * If an invalid email address is entered, verify that Bob is prompted to re-enter it and an account isn’t created. * If the passwords don’t match, verify that a message tells bob this and an account isn’t created. * If all of the details are valid, verify that an account is created and Bob is taken to the email verification page. |
|  | Bob wants to enter his card details so that he can order a taxi in the future. | * If a field is left empty, verify that an error message appears and Bob has to re-enter his details. * If an invalid card number is entered, verify that Bob is prompted to re-enter it. * If the details are correct, verify bob is taken to the taxi confirm page. |
|  | Derek wants to order a taxi so he can go to work. | * If Derek has not entered his card details, verify he is prompted to enter them first. * If there is a problem with the transaction, verify Derek is made aware of this. * If the transaction goes through, verify that it is confirmed, and the details of the route will be made available. |
|  | Derek wants to view his journey | * After ordering the taxi, verify that Derek is notified the taxi is on its way. * When the taxi has reached Derek, verify that he is notified that the taxi is here. * When the destination has been reached, verify Derek has been notified. * After Derek has finished his journey, verify that he can review the driver. |

### Architectural designs

Table

Description automatically generatedLayered Architecture design 1:

The first layer has user interaction, form validation, and an introduction to the website, as these are what will interact with the user. Below that is the UI management layer, which ensures that the first layer can work properly, by authenticating users and so on. The third layer will set up a ride, which is the main functionality of our application. The fourth will manage the ride that has been set up in the previous layer. We will use some basic services such as a map API for our map services.

Server Distribution Architecture 1:

Diagram

Description automatically generated

Firstly, we have decided to have three separate servers, the web server, application server and the database. This is so that we can manage each of them effectively, and it also helps hosting them, as there are no free services that will host all three servers. We also have a load balancer to distribute the network traffic so that our server will not be overloaded.

Microservices Architecture 1:Diagram

Description automatically generated

Microservices Architecture 1 - We have decided to decompose the driver and user components into smaller microservices so that they can be easily managed and updated. The website will connect to a rest API which will have two hub routes for drivers and users, and then further subroutes for every microservice. Every microservice in each component will share a database, e.g. all the driver services use the driver db. This is because driver and user data should all be stored in the same database to ensure that our system is intuitive to use.

### Use Case

Diagram

Description automatically generated

### Technology Choices

|  |  |  |
| --- | --- | --- |
| Component | Technology Choice | Justification |
| Database | MongoDB(NOSQL) | Firstly, MongoDB is a technology that the members of our group are most familiar with working with. In addition, it is the most suitable for our system, as there is a lot of unstructured data to store regarding the user and driver. |
| Platform | Web Platform | All of our team are very well-versed in web development, meaning that we already had the skills available to make a website that works on both phone and desktop. In addition, development of a website goes hand in hand with the API and backend as we can work on these in the same development environment, working with node js and express. |
| Server | Public cloud | As we are a small company, we do not have the resources to set up and manage our own servers, so we will be using the cloud to store our services. We are using Amazon web service to store the database, as this is where our user data will be stored, so a secure, reliable service is required. The frontend will be hosted on Netlify, and the API will be hosted on Heroku. |
| Open source | OpenStreetMap, geoLocation | We are using the W3 standard geoLocation library which is as standard implemented into JavaScript. This is used for grabbing the user location which is necessary when dealing with locations of customers and drivers, it works with longitude and latitude for simplicity and accuracy.  Due to this simplicity it is also possible to pass the longitude and latitude values to another api such as OpenStreetMap which has been considered as alternative to google maps due to it being a free to use service. |
| Development Tools | Visual Studio Code | Due to our architecture being a very service-oriented website, we will be developing our system in visual studio code as it complements this style of development. Visual studio code is often used in web development as it offers a lot of plugins that are useful for creating a website. |

### Reasons for using software as a service

Drivers:

* We have no infrastructure in place to run the servers
* Not enough time before the deadline to set up any servers of our own
* The cost of setting up a server ourselves is too high
* Fluctuating demands (busier on Friday nights than other times)

Solutions

* Using MongoDb atlas to store our database
* Using Netlify to host our web server
* Using Heroku to host the backend API

Benefits

* Allows for us to start development instantly without waiting for servers to be set up
* As we are a small business, there are no costs for any of the services used
* Our infrastructure is very scalable if we get more traffic in the future
* We can unsubscribe or level down from services at times of lower demand

### Testing

Table

Description automatically generated

Table

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

\*Test plan is in screenshot format to avoid word count being counted twice for more info please see the text version which is filled in below.

#### **Evidence of acceptance testing based on acceptance tests derived from user stories.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario Tested | Test Data | Expected Outcome | Actual Outcome | Actions Taken |
| If a field is left empty during account creation. | Empty password field. | User account is not created until all account details have been entered. |  |  |
| If an invalid email address is entered during account creation. | Check if email is the correct format of name@company.com. | Detects invalid email format and prompts the user to try again before account creation is successful. |  |  |
| If the passwords don’t match when confirming password for account creation. | Passwords in two fields are different.  Password  Password1234 | Stops account creation with a prompt until both password and confirm password match. |  |  |
| If all of the details are valid when creating an account. | All valid expected inputs provided.  KieranUser  email@outlook.com  Password1234  Password1234 | User account is created and the user is taken to the home page/dashboard. |  |  |
| If a field is left empty when entering card details to be stored. | Card number field empty. | An error message is output and the user has to re-enter their details. |  |  |
| If an invalid card number is entered when entering card details to be stored. | Letters in card-number field.  Not long enough numbers in the card-number field.  abcdefg  1234-0000 | User is prompted to re-enter the card number until a valid one is input which can then be stored. |  |  |
| If the card details are correct when entering card details to be stored. | All valid expected inputs provided.  Kieran Foy  1234-5678-9123-4567  12-23  345 | Card details are stored and the user is taken to a confirmation page or given a notification of success. |  |  |
| In the case of a user attempting to book a taxi with no card details stored. | No card details are entered. | Software should prevent taxis being ordered without first asking for card details. |  |  |

### How formative feedback was used to improve our system

Sprint 1

The first sprint was based on the overall design of the system. We wanted to make sure that the overall look and feel of the system was what the client needed before we moved on with the more technical requirements.

|  |  |
| --- | --- |
| Feedback Given | Action Taken |
| Try to make sure that the layout of the app resembles solutions already on the market, so that users will be familiar with the interface. | The design was edited so that it resembles Uber, the most popular taxi app. This was done by having a home screen containing a map and the ability to order a taxi, as well as a menu that appears from the side when needed. |
| Have as much functionality as possible on one page to avoid confusion of navigating through different tabs. |  |

Sprint 2

The second sprint focused on the user side of things. In this sprint, the ability to create an account, log in, log out, and input the user’s card details were added. These features were chosen to be in this sprint as we thought it best to have the basic user functions complete before moving on to the taxi specific functionality.

|  |  |
| --- | --- |
| Feedback Given | Action Taken |
|  |  |
|  |  |

Sprint 3

The third sprint involved the main use of the system, such as the map, the ability to book a taxi, and allowing for the user to review a driver. These were chosen to be added last as we wanted to have a strong base of a system before adding in the more technical features, as the taxi booking and map functionality were the most difficult to implement.

|  |  |
| --- | --- |
| Feedback Given | Action Taken |
|  |  |
|  |  |

### References

mozilla. “GeolocationCoordinates.longitude.” *MDN Web Docs*, Mozilla, https://developer.mozilla.org/en-US/docs/Web/API/GeolocationCoordinates/longitude. Accessed 2nd November 2021.

Mozilla. “Using the geolocation API.” *MDN Web Docs*, Mozilla, https://developer.mozilla.org/en-US/docs/Web/API/Geolocation\_API/Using\_the\_Geolocation\_API. Accessed 2nd November 2021.

w3schools. “Html5 geolocation.” *w3schools*, w3schools, https://www.w3schools.com/html/html5\_geolocation.asp. Accessed 2nd November 2021.

# Peer Assessment form

|  |  |  |
| --- | --- | --- |
|  | Team member + work done | Mark out of 10 |
| 1 | Domonic Cassidy   * User stories * Back-end dev (database, data handling, etc) * Architectural design * Technology choices * SAAS reasons | / 10 |
| 2 | Haider Sheikh   * User stories * Front end design (CSS, HTML, etc) * Architectural design | / 10 |
| 3 | Kieran Foy   * User stories * Front end map functionality(JS, design, etc) * Use case diagram * Test plan & test data | / 10 |
| 4 | Nathan Yianni | / 10 |
| 5 |  | / 10 |