Requirements traceability

*1.1: The system must be accessible by the passengers through the website and the mobile application*

This requirement can be satisfied thanks to MTSPassengerWebController and its higher components WebBrowser and MTSPassengerWebView (in case of website) and MTSPassengerMobileController and its higher component MTSPassengerMobileView (in case of mobile application).

*1.2: The system must allow passengers to create an account*

This requirement can be satisfied thanks to MTSModel, MTS\_DB and all their higher components on the side of the passenger.

*1.3: The system must allow passengers to log in*

This requirement can be satisfied thanks to MTSModel, MTS\_DB and all their higher components on the side of the passenger.

*1.4: The system must allow passengers to edit their account information*

This requirement can be satisfied thanks to MTSModel, MTS\_DB and all their higher components on the side of the passenger.

*1.5: The system must provide a form in order to allow passengers to make a request*

This requirement can be satisfied thanks to MapsServer and all its higher components on the side of the passenger.

*1.6: The system must allow the passenger to check the current state of his request (phase of processing, driver’s location, accepted request information)*

This requirement can be satisfied thanks to MapsServer and all their higher components on the side of the passenger.

*1.7: The system must allow passengers to cancel a request*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the passenger.

*1.8: The system must inform the taxi driver (and the other passengers if necessary) when he receives a cancelation*

This requirement can be satisfied thanks to EmailServer (in case is not possible to take contact with the passengers) and all its higher components on both sides (the passenger’s one and the taxi driver’s one).

*1.9: The system must delete the request for service information, incoming request information and accepted request information ten minutes after the estimated arrival time*

This requirement can be satisfied thanks to MTSModel and all its higher components on both sides (the passenger’s one and the taxi driver’s one).

*1.10: The system must not allow any passenger to make more than one request*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the passenger.

*1.11: The system will send notifications to the passenger wherever he is logged in (mobile application or web site). If he is logged in nowhere, the notification is sent via email*

This requirement can be satisfied thanks to EmailServer (if he is not logged in) or MTSModel (if he is logged in) and all its higher components on the sides of the passenger.

*2.1: The system must be accessible by the taxi drivers through a mobile application only*

This requirement can be satisfied thanks to MTSTaxiDriverMobileController and its higher component MTSTaxiDriverMobileView.

*2.2: Taxi drivers must be able to register to the application*

This requirement can be satisfied thanks to MilanoGovernment and all its higher components on the side of the taxi driver.

*2.3: The system must allow registered taxi drivers to login*

This requirement can be satisfied thanks to MTSModel, MTS\_DB and all their higher components on the side of the taxi driver.

*2.4: The system must allow taxi drivers to edit their account information*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the taxi driver.

*2.5: The system must provide an availability section allowing drivers to change their status*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the passenger.

*3.1: When a taxi driver receives an incoming request, the system must allow him to see the incoming request information*

This requirement can be satisfied thanks to MapsServer and all its higher components on the side of the taxi driver.

*3.2: The system must allow taxi drivers to accept an incoming request*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the taxi driver.

*3.3: When a taxi driver accepts an incoming request, the system must change is status to not available*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the taxi driver.

*3.4: When a taxi driver accepts an incoming request, the system must send to the passenger the accepted request information*

This requirement can be satisfied thanks to EmailServer (if passenger is not reachable otherwise) or MTSModel and all its higher components on the side of the passenger.

*3.5: When a taxi driver has accepted an incoming request, the system must allow him to see in a map his current location and the passenger(s) origin(s) and destination(s)*

This requirement can be satisfied thanks to MapsServer and all its higher components on the side of the taxi driver.

*3.6: The system must allow taxi drivers to decline an incoming request*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the taxi driver.

*3.7: When a taxi driver declines an incoming request, the system must resend it to the next available passenger*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the taxi driver.

*3.8: When no available taxi drivers are found, the system must cancel the request and inform it to the passenger(s)*

This requirement can be satisfied thanks to EmailServer (if passengers are not reachable otherwise) or MTSModel and all its higher components on the side of the passenger.

*3.9: The system must allow taxi drivers to cancel an already accepted request*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the taxi driver.

*3.10: When a taxi driver cancels an already accepted request, the system must inform it to the passenger*

This requirement can be satisfied thanks to EmailServer (if passenger is not reachable otherwise) or MTSModel and all its higher components on the side of the passenger.

*4.1: The system must be able to recognize the requests with correct request for service information*

This requirement can be satisfied thanks to MTSModel.

*4.2: The system must be able to process the recognized requests*

This requirement can be satisfied thanks to MTSModel.

*4.3: The system must use zone queues to manage available taxi drivers*

This requirement can be satisfied thanks to MTSModel.

*4.4: The system must send a processed request only to the first available taxi driver in the zone queue that corresponds to the origin of the request*

This requirement can be satisfied thanks to MTSModel.

*4.5: The system must use the taxi’s GPS to know the position of the driver*

This requirement can be satisfied thanks to MapsServer, MTSIntegration and MTSModel.

*4.6: The system must use the Map service to assign a zone queue to the driver according to its position*

This requirement can be satisfied thanks to MapsServer, MTSIntegration and MTSModel.

*4.7: When the system has identified the zone of a just available taxi driver, it must send him to the bottom of the zone queue*

This requirement can be satisfied thanks to MTSModel.

*4.8: The system must refresh the zone of every available taxi driver every 5 minutes*

This requirement can be satisfied thanks to MapsServer, MTSIntegration and MTSModel.

*4.9: When an available taxi driver changes his zone, the system must send him to the bottom of the zone queue*

This requirement can be satisfied thanks to MTSModel.

*4.10: The system must send to the bottom of the zone queue those taxi drivers who decline a received request*

This requirement can be satisfied thanks to MTSModel.

*4.11: The system must send to the bottom of the zone queue those taxi drivers who cancel an already accepted request*

This requirement can be satisfied thanks to MTSModel.

*4.12: The system must send to the top of the zone queue those taxi drivers that have received a cancelation for an already accepted request*

This requirement can be satisfied thanks to MTSModel.

*4.13: The system must remove from of the zone queue those taxi drivers that change their status to not available*

This requirement can be satisfied thanks to MTSModel.

*5.1: The system must provide a form in order to allow passengers to make a request*

This requirement can be satisfied thanks to MapsServer and all its higher components on the side of the passenger.

*5.2: The option “Taxi sharing” will be effectively activated only if there will be other people who have requested it in the same area at the same time*

This requirement can be satisfied thanks to MapsServer and all its higher components on the side of the passenger.

*5.3: Passenger will receive a notification by the system in which there will also be specified if the taxi sharing option has been effectively accepted*

This requirement can be satisfied thanks to EmailServer (if passenger is not reachable otherwise) or MTSModel and all its higher components on the side of the passenger.

*6.1: The system must provide a form in order to allow passengers to make a request*

This requirement can be satisfied thanks to MapsServer and all its higher components on the side of the passenger.

*6.2: The reservation must be made by the passenger at least 2 hours before time of departure*

This requirement can be satisfied thanks to MTSModel and all its higher components on the side of the passenger.

*6.3: The system has to confirm the reservation to the passenger*

This requirement can be satisfied thanks to EmailServer (if passenger is not reachable otherwise) or MTSModel and all its higher components on the side of the passenger.

*6.4: The system allocates a taxi 10 minutes before the departure time in the requested place*

This requirement can be satisfied thanks to MTSModel.