**RESTful Architecture for Strelsau’s Taxi Ordering System**

Traditionally, the taxi market in Strelsau (Ruritania) has been highly fragmented. As of 2012, there were around 20 taxi companies, each one operating an average of 100 taxis. Each company handled its taxi ordering service inefficiently based on rudimentary systems used by operators at the call centres of the companies.

The situation is about to change with the launch of United Taxis of Strelsau (UTS), which will absorb about half of Strelsau’s taxi companies. United taxis will manage a fleet of around 1000 taxis, with the ambition to grow up to 1500 taxis by 2015.

United Taxis will put in place a single taxi ordering system: The Strelsau Taxi Ordering System (STRS). STRS will be multi-channeled. Customers will still be able to order a taxi via phone (the traditional way), but additionally, they will be able to do so via a Web-based front-end and via mobile applications available for all major smartphones (Android, iPhone and Windows 8). The Web front-end will allow customers to enter the address where they require a taxi, their name and their mobile number. STRS will inform the customer of the time when the taxi will pick them up. This information will also be sent to the customer via SMS. Updates will also be sent to the customer in case the taxi is delayed.

When reserving via a smartphone application, the application will automatically fetch the GPS location of the customer (if available), so that that the customer does not need to enter the location explicitly – assuming they wish to be picked from their current location. Customers will instantly get an estimate of how much time the taxi will take to pick them up.

All taxis will be equipped with Android tablet PCs with GPS. An application called TaxiHome will be installed in each of these devices. Taxi drivers will use TaxiHome to notify STRS of their availability. At a given point in time, a taxi can be off-duty, available, busy and invisible (this status is taken when no status update has been received by STRS for more than 5 minutes). When taxis are in available or busy states, TaxiHome will periodically report the location of the taxi to STRS.

When STRS receives a new taxi order, it assigns it to the closest available taxi. If there are several taxis at equal or almost equal distance from the location, the taxi that has been available the longest is assigned (i.e. first-in-first-out). STRS will communicate the assignment to the corresponding taxi. STRS will be built using a location service called Awaze, which is able to calculate the distance and approximate travel time between any two locations in Strelsau, given as input the GPS coordinates or the address identifiers of the locations in question, and taking into account roadworks and traffic. Once an order has been assigned to a taxi, the taxi's terminal is
notified that an order has been assigned to them via Google Cloud Messaging for Android (GCM). The taxi driver can retrieve the details of the order via TaxiHome, and accept the order or reject it. If rejected, an alternative taxi is assigned by STRS.

**Task 1 [15 points]** Design a service-oriented architecture for STRS. This architecture should include the actors of the system, any mobile applications and Web applications, the Web services forming the backend of STRS, and any other relevant system components. Please present the architecture in the form of three diagrams:

- An entity diagram (e.g. UML class diagram) showing the key data entities that the STRS system needs to manage, and their main attributes and associations.
- An interaction diagram where the arrows denote possible interactions and the label in each arrow indicates the purpose of the interaction. This diagram should be business-oriented, meaning that it should allow people in the technical team to communicate with business stakeholders at UTS.
- A layered architecture diagram (3 or 4-tiered) where the arrows denote "uses" relations (e.g. a service "uses" another service). This diagram is intended for communication within the IT team and it can use technical jargon.

Your diagrams should be as self-explanatory as possible, but you can include additional textual explanations as you deem appropriate.

**Task 2 [20 points]** Specify a RESTful interface for the backend services identified in the previous task. For each operation provided by the backend, you need to specify:

- The name of the operation
- A brief explanation of what the operation does
- A list of input and output parameters of the operation. For each parameter, give a name, a brief explanation of the meaning of the parameter, whether the parameter is “required” or “optional”, and a description of how the parameter will be encoded, i.e. whether the parameter will be transferred using URL tunnelling (form-encoded) or in the body of the request/response, and in the latter case, what Internet media type (Content-type) will be used.
- The relative URL where the operation will be available.
- Whether the operation uses the GET, POST, PUT or DELETE method.
- A list of HTTP status codes that the operation may return and for each code, 1 or 2 sentences explaining under what conditions this status code is returned.

**Task 3 [15 points]** Specify a set of Java classes to implement the RESTful API using JAX-RS and JAXB. Skeletons should given for Java classes corresponding to resources (JAX-RS) as well as for Java classes corresponding to documents (JAXB). The skeleton of each class should include:

- The class name and JAX-RS or JAXB annotations associated with this class
- For each method, method name, input parameters and return type, JAX-RS or JAXB annotation associated with the method and JAX-RS or JAXB annotation associated with each parameter (where applicable).
- For each attribute, attribute name, type and JAXB annotation if applicable.

**Task 4. [3 bonus points]** Taxi operators pay a fee to UTS equivalent to 3% of the revenue generated by orders made via STRS. Extend the RESTful interface of Task 2 to collect the data needed for billing. Do not write Java classes for this task.