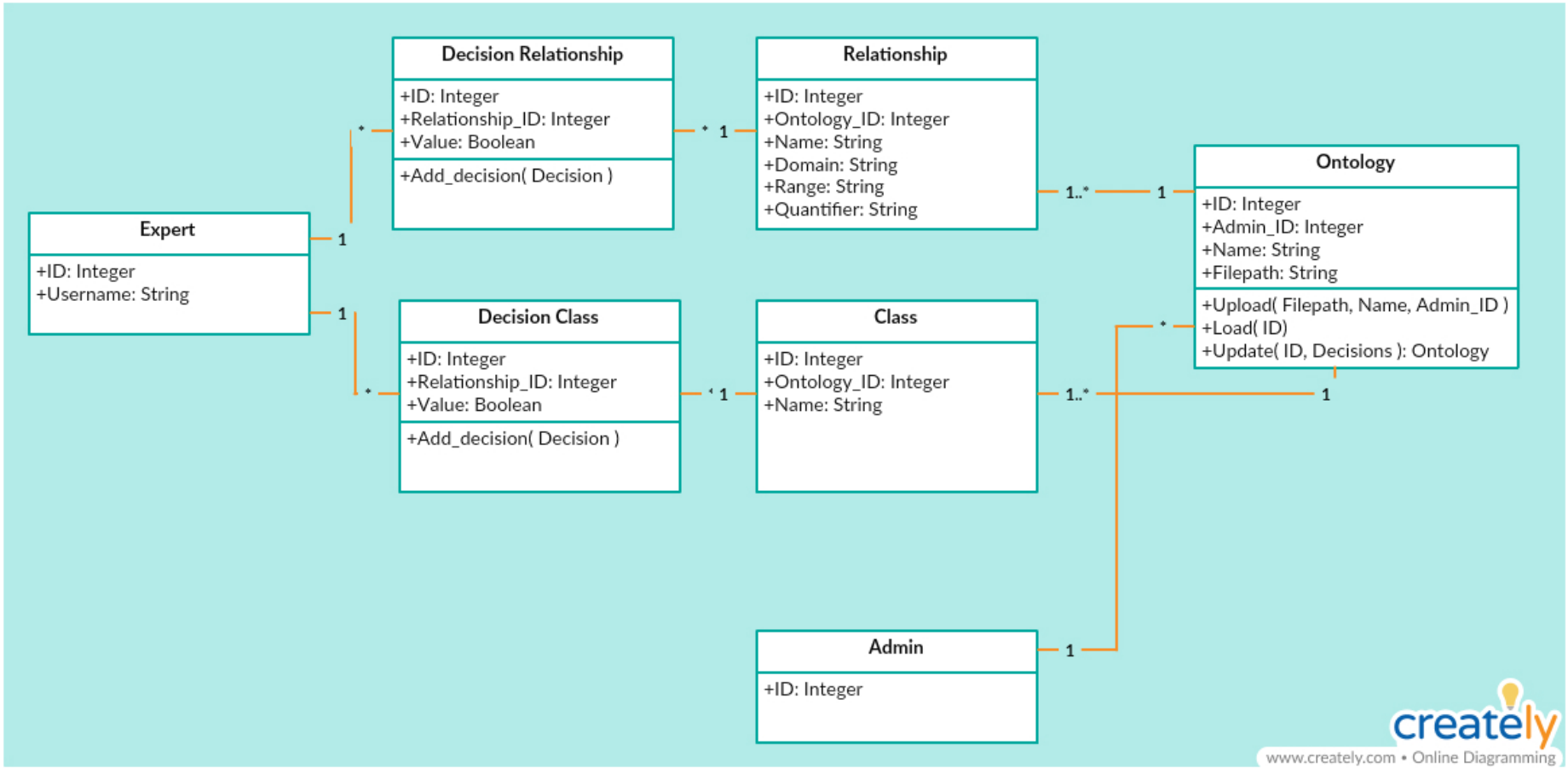
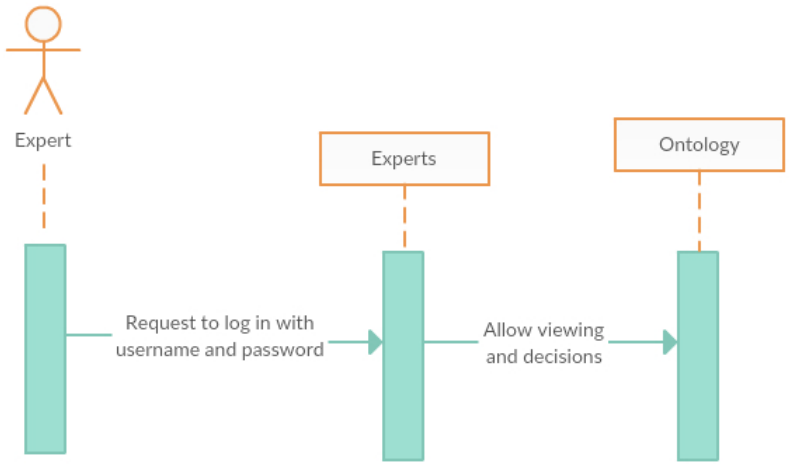



| | |
|---|---|
| **Expert** | **Class state**<br>• ID: Unique identifier<br>• Username: Identifier for authentication |

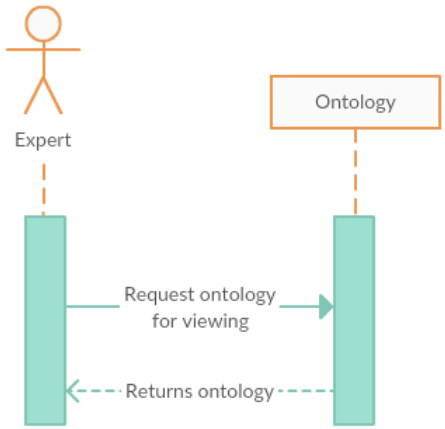
| | |
|---|---|
| **Ontology** | **Class state**<br>• ID: Unique identifier<br>• Admin_ID: ID of owning admin<br>• Name: Name of ontology for viewing<br><br>**Class behaviour:**<br>• Upload: Add new ontology on server<br>• Load: Retrieve ontology for viewing<br>• Update: Generate new ontology file based on decisions |
| **Relationship** | **Class state**<br>• ID: Unique identifier<br>• Ontology_ID: ID of ontology to which the relationship belongs<br>• Name: Name of relationship<br>• Domain: Domain on which relationship is defined<br>• Range: Range of values the relationship can relate<br>• Quantifier: Type of restriction on quantities |
| **Decision** | **Class state**<br>• ID: Unique identifier<br>• Relationship_ID: ID of relationship which is decided<br>• Value: Boolean indicating whether decision was accepted<br>**Class behaviour**<br>• Add_decision: Add a new decision to the database |
| **Admin** | **Class state**<br>• ID: Unique identifier |

***Sequence Diagram(s):***

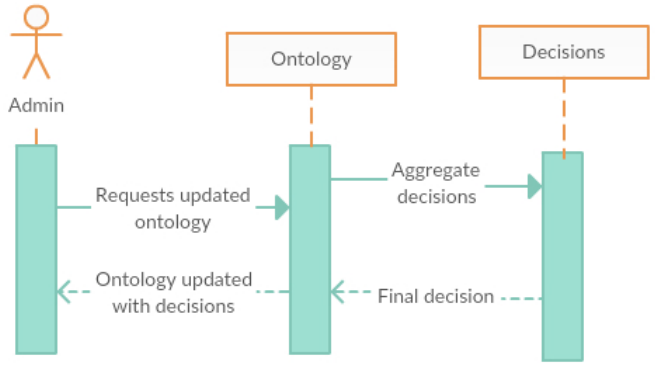
Authenticate:



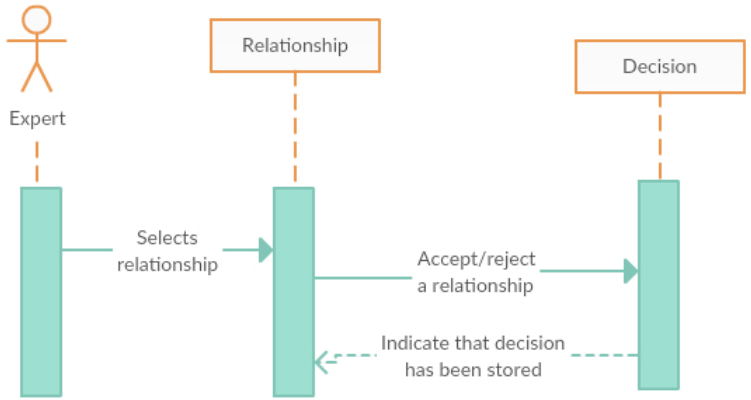

Load and Visualise:



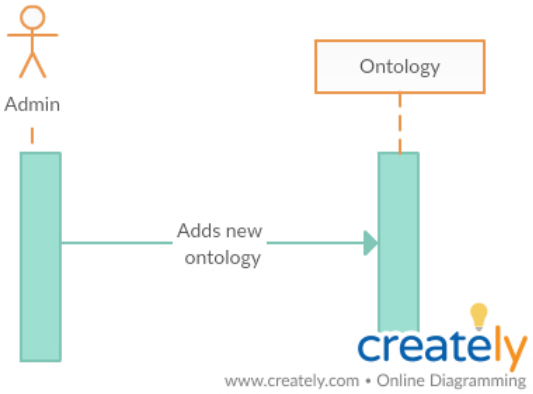

Accept/Reject for relationships:



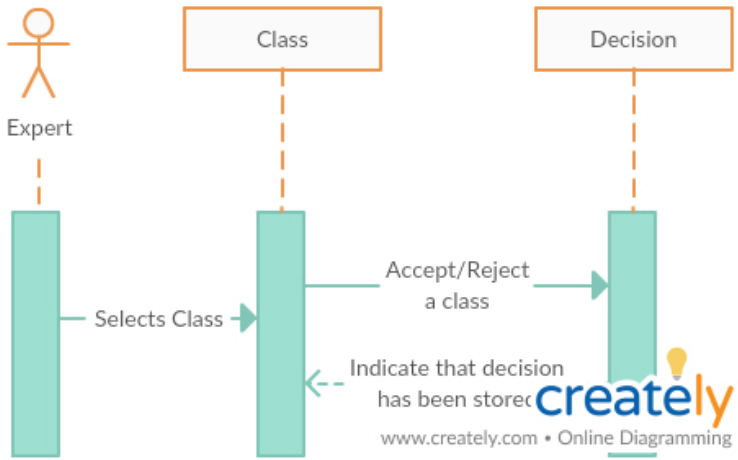

Get verified ontology:




Store ontology:



Accept/Reject for classes:




***Design Rationale:***

*We had an option between two visualisation softwares - Ontograf for Protege, and WebVOWL. Ontograf is a plugin for a desktop software, and if it had to be used, it it would have to be made compatible with a web application. WebVOWL was a standalone visualisation package, and an ontology management system had to be built. We decided to use WebVOWL since we believed a good visualisation system would be more difficult to build than an ontology management system.*

*We also decided to use OWLAPI for writing changes ( deletion of restrictions, subclass axioms, classes ) to the owl file since OWLAPI had good documentation available and there was a lot of functionalities for performing basic tasks like loading an ontology, deletion of axioms, saving an ontology file and many other features. Jar files were used since it is an easy way to implement an executable, we just had to send the IRI's of the classes to these jar files as arguments and it would execute the respective deletion task.*