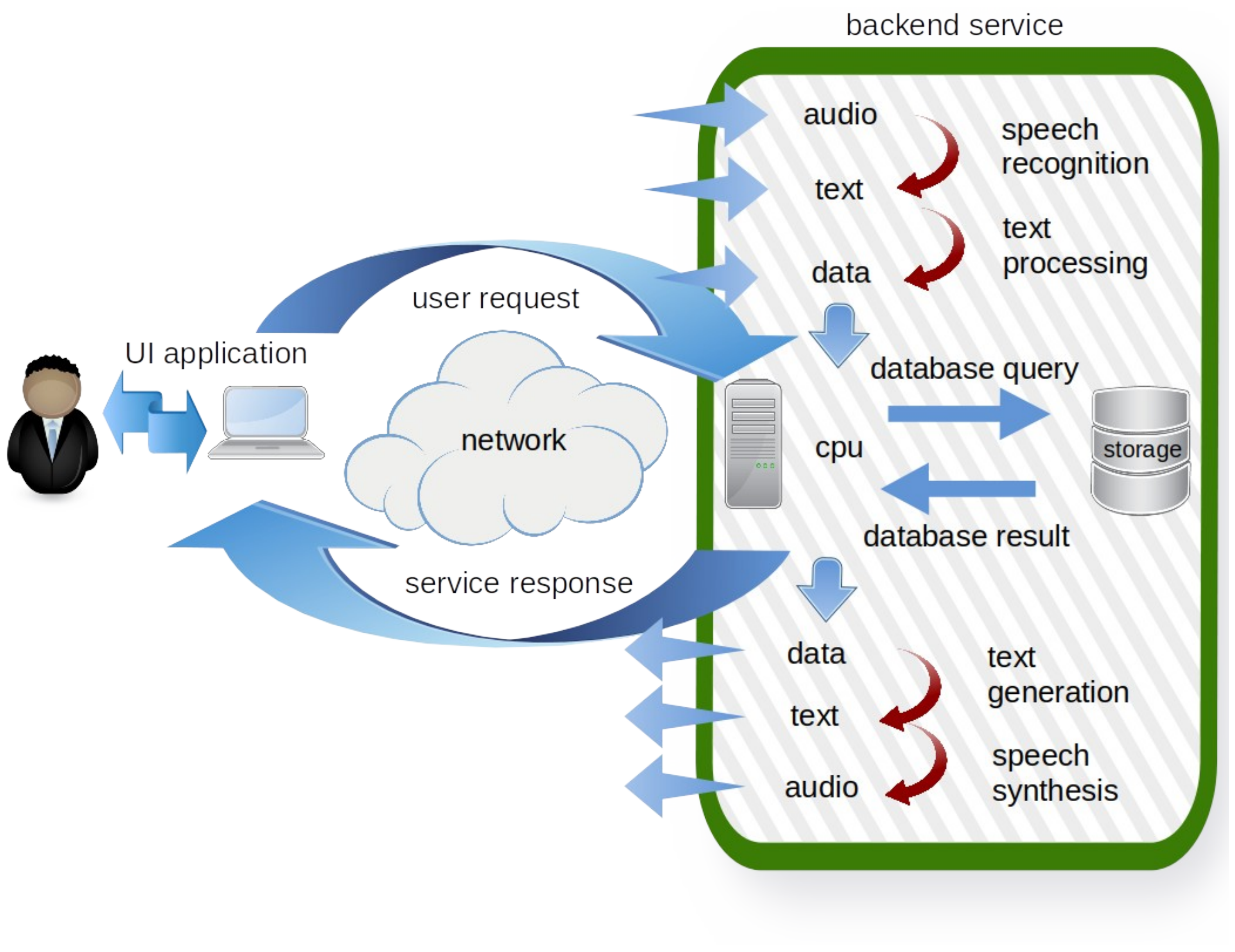
Conversational backend service

This is a draft specification for building a conversational UI backend service. The service described here uses a modular architecture to implement all its features without any vendor lock-in. Each service module can be implemented with online services or locally installed open/close source software to provide the desired data processing features.



Non-exhaustive list of possible module implementations

speech recognition module (audio input to text output)
- AWS Transcribe https://aws.amazon.com/transcribe/
- CMU Sphinx https://cmusphinx.github.io/
- Jarvis Speech API https://github.com/lkuza2/java-speech-api

text processing module (text input to data query output)
- AWS Lex https://aws.amazon.com/lex/

- AWS Comprehend https://aws.amazon.com/comprehend
- Apache OpenNLP https://opennlp.apache.org/
- Stanford CoreNLP https://stanfordnlp.github.io/CoreNLP/

database query module (data query input to query result output)
- this module receives parsed text into structured data that can query a database
- it depends of text parsing capabilities to generate the proper query expression
- the most likely scenario is to have a generic REST API that forwards the request to any supported database backend
- there is no turn-key solutions for this module as it is usually dependent on the data model
- it is probably the only module we would have to build in house
- a first prove of concept could be to use the TMDB API to try automated translation
  of text examples at https://www.themoviedb.org/documentation/api/discover
- we can also try the same questions in the link above on an E2 data model

database result module(query result input to data format output)
- formatters for any supported response mimetype format

text generation module (query result input to text output)
- AWS SageMaker https://aws.amazon.com/sagemaker/
  https://www.kevinhooke.com/2018/07/19/using-aws-sagemaker-to-train-a-model-to-generate-text-part-1/
- KPML http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/kpml-description.htm
- SimpleNLG https://github.com/simplenlg/simplenlg
- many more tools and data at https://aclweb.org/aclwiki/Natural_Language_Generation_Portal

speech synthesis module (text input to audio output)
- AWS Polly https://aws.amazon.com/polly/
- FreeTTS https://freetts.sourceforge.io/
- Jarvis Speech API https://github.com/lkuza2/java-speech-api

simple REST API example the service can expose:

/input
- receives user query and uses content negotiation to know how to interpret it
  - "Content-type" HTTP header tells the service what is the data format of the request
  - "Accept-Language" HTTP header tells the service the natural language the user is speaking (french, english, spanish, ...)
  - "Accept" HTTP header tells the service in what format the user wants the response
- if content type is audio/wav then call the speech recognition module
- if content type is text/plain then call the text processing module
- if content type is application/json then call the database query module
- other content types could be supported if corresponding modules are installed
- correspondingly, accept header values of audio/wav or text/plain will have the service
  process the database query result through text generation and speech synthesis modules
  before sending the response back to the caller

/data
- should expose the data model and content that every module will use
- a user could manage database content directly with only REST methods if he has appropriate permissions

/history
- contains the logs of all user requests and service responses sorted by timestamp

/service
- exposes what implementations are available and in use for each module
- with apropriate permissions, a user can change the selected implementation for a specific module
- the change can be applied for the current session only, for the current user or as the default for all users
- this will be useful for automated validation testing of any module implementation