**Survey Builder Backend**

Technology used:

-Mongo DB

-Node js

-Rest service

Description:

-With node js we are making connection to MongoDB; to insert /fetch/update/delete documents from Frontend

-For survey, we are using survey Schema

-For response, we are suing response schema

Survey Schema:

Survey schema will be in form:

{

"SurveyNo" : "<>", // to uniquely identify a survey

"SurveyTitle" : "<>", //any title of survey

"CreatedBy" : "<>", //logged in user who is creating the survey

"Questions" : [

{ "QuestionType" : <>, //question type is 1 for text based, 2 for single option correct, 3 for multi option correct, 4 for matrix likert

"QuesionNo" :<>,

"QuestionText" : "<>", //question

"Options" :[ //options is list of option ; list be null if text based question, but multiple objects in case of question type as 2 3 4

{

"name":"<>", //name will store the option ,and levels will store scale only for matrix likert question type else will be null

"levels" : []

}

]

}

]

}

Example 1:

{

"SurveyNo" : "5",

"SurveyTitle" : "IIITH zoom call meetings",

"CreatedBy" : "surveydeveloper2@iiit.ac.in",

"Questions" : [

{ "QuestionType" : 1,

"QuesionNo" :1,

"QuestionText" : "How much rank you got in gate"

},

{ "QuestionType" : 2,

"QuesionNo" :2,

"QuestionText" : "How will you rate zoom meetings?",

"Options" :[

{

"name":"Good"

},

{

"name":"Avg"

},

{

"name":"Excellent"

},

{

"name":"V Excellent"

}

]

},

{ "QuestionType" : 3,

"QuesionNo" :3,

"QuestionText" : "Which part(s) you like the most in zoom meetings?",

"Options" :[

{

"name":"speed"

},

{

"name":"picture quality"

},

{

"name":"latency"

},

{

"name":"buffering"

}

]

},

{ "QuestionType" : 4,

"QuesionNo" :4,

"QuestionText" : "IITH zoom meetings quality subject wise..give rating",

"Options" :[

{

"name":"SSD",

"levels" : ["4","3" , "2" ,"1"]

},

{

"name":"AOS",

"levels" : ["4","3" , "2" ,"1"]

},

{

"name":"DSA",

"levels" : ["4","3" , "2" ,"1"]

},

{

"name":",MATHS",

"levels" : ["4","3" , "2" ,"1"]

}

]

}

]

}

Survey schema has various rest endpoints to interact with mongo DB :-

POST surveys from fronted to backend

/add\_survey

GET all surveys from backend to frontend

/surveys

GET surveys by logged in user and survey number from backend to frontend

/surveys/:devmail/:SurveyNo

GET surveys by logged in user from backend to frontend

/surveys/:devmail

DELETE survey

/delete/survey/:SurveyNo

UPDATE a survey

/update/survey/:SurveyNo

Response Schema:

Response schema will be in form:

{

"SurveyNo" : "<>", // to uniquely identify a survey

"CreatedBy" : "<>", //logged in user who is creating the survey

"Participant": "<>", //particpant who is actually giving the survey

"Response" : [ //Response will be list depending on how many questions survey had

{ "QuestionNo" :<>,

"QuestionText" : "<>",

"Ans" : [] //list of answer(s) , will have 1 value for text based or single choice option question , but can be multiple for question type 3 and 4

}

]

}

Example (response to above survey on zoom meetings)

{

"SurveyNo" : "5",

"CreatedBy" : "surveydeveloper2@iiit.ac.in",

"Participant": "shaon.dasgupta@iiit.ac.in",

"Response" : [

{ "QuestionNo" :1,

"QuestionText" : "How much rank you got in gate",

"Ans" : ["1001"]

},

{

"QuestionNo" :2,

"QuestionText" : "How will you rate zoom meetings?",

"Ans":["Good"]

},

{

"QuestionNo" :3,

"QuestionText" : "Which part(s) you like the most in zoom meetings?",

"Ans":["speed" , "latency"]

},

{

"QuestionNo" :4,

"QuestionText" : "IITH zoom meetings quality subject wise..give rating",

"Ans":["4","3","4","1"]

}

]

}

Response schema has various rest endpoints to interact with mongo DB :-

POST responses from fronted to backend

/add\_response"

GET all responses from backend to frontend

/responses

**Transcript Generator:**

* In the Transcript generator, the developer can upload an audio/video file of any format, containing an interview recording and then generate and download the transcript of the recording in a text file.

**Implementation:**

* The main code for converting the audio file to the text is written in python language.
* The transcript generator is implemented as follows:
  + **Front end:**
    - Developer uploads a video/audio file from the browser.
    - The video/audio file will be sent to the back-end(express js server) in the form of a Buffer Object.
  + **Back end:**
    - In the back-end code, the Buffer Object is converted back to the video/audio file.
    - The back-end server makes use of the spawn() method and runs the python code (internally on the terminal) on the server.
    - The python code converts the video file into an audio file using moviepy library and then divides the audio file into chunks.
    - Then, the text for each chunk is generated using the SpeechRecogniton library available in python. It internally uses recognize\_google() method to convert the audio chunk into text.
    - Finally, the text generated for each chunk is combined and sent back to the browser.
  + In the front-end, the text response is stored in a .txt file and made available to the user for downloading.
  + The accuracy and quality of text generated depends on the input audio/video file and the SpeechRecognition library.

**Server requirements:**

* Python requirements:
  + Install SpeechRecogntion and moviepy packages.
    - command: *pip install SpeechRecogintion*
    - command: *pip install moviepy*
* Node JS requirements:
  + Install express-fileupload package.
    - command: *npm install --save express-fileupload*