A picture containing text, plant, dark

Description automatically generated**INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY**

**PROJECT REPORT**

***SURVEY BUILDER AND TRANSCRIPT GENERATOR***

Guided by:- Submitted by:- (BRAINSTORMERS)

Saianirudh Karri Shaon Dasgupta(20212010xx)

Arvind Narayanan(TA) Vishal Mani Mishra(2021201050)

Yaswanth Haridasu (2021201032)

Gargi Dwivedi (20212010xx)

Ritabrata Podder(2019900031)

LINKS

**Project GitHub Repository URL:** <https://github.com/Rito2323/SSD-project-Team7>

Demo Video Link

INTRODUCTION

About The Project: -

This Project consists of two parts, survey builder and Transcript Generator. In the Survey Builder following tasks can be performed: -

1. Researchers can login to create, modify and upload a survey form. This survey form link can be shared among participants.
2. Participants can login and answer the questions asked in the survey form.
3. Responses of participants can be seen by researchers whenever he/she wants.

In Transcript Generation part, after login, researcher can upload the video or audio file in the respective field and generates the transcript file and download it in the end.

REQUIREMENTS

1. Stakeholders

- PhD Students of IIIT, Hyderabad.

- MS Students of IIIT, Hyderabad.

- Research Faculties of IIIT, Hyderabad.

- Focussed Interns

- Other survey participants.

2. Roles and Responsibilities of Stakeholders

- MS and PhD students and faculties (Survey Developers) They will have the role as the survey developers and will use the survey builder to create and publish new surveys. Also, they may use the Transcript generator to generate the transcript of an interview. After publishing the form, they can view the responses in the response section.

- Focussed Interns and other survey participants

They will fill the published survey and submit their responses which would be recorded and viewed by the survey developer.

**Workflows**

Survey Developer:

Workflow 1: Survey Builder

The developer will be able to see the Entry Dashboard UI. On selecting the Survey Builder option in the Dashboard, the user would be navigated to the survey builder page.

In the survey builder page, the developer would be able to select the type of question based on the response type and add the question to the survey question list.

After the questions are selected, the user should be able to preview his questions and can reorder them or go back to editing them.

If the preview is satisfactory the user can save and publish the survey. The survey form data will be converted to json format and stored in the DB. Also, publishing the survey would generate a survey link that the developer can share with the participants.

Also, developers can delete existing surveys or update existing surveys.

Workflow 2: Response Viewer

After creating and publishing the survey, the developer should be able to view and analyze the responses by navigating to the Responses UI from the dashboard.

Under the Responses UI, the responses of all the participants in each survey created and published by the developer could be viewed.

Workflow 3: Transcript Generator

From the dashboard, the developer should be able to navigate to the Transcript generator to generate a transcript of an interview recording.

In the Transcript generator, the developer should be able to upload an audio/video file of any format, containing an interview recording and then generate and download the transcript file.

Participant:

Workflow 1: Survey fill up and submission

The participant who has access to the survey link should be able to fill it up and submit the responses.

The responses submitted should be stored in the DB.

**TECHNICAL DETAILS**

**Survey Builder Frontend**

**Technology Used**

* React JS
* Redux
* Html
* CSS
* JS
* Bootstrap

**Description**

Using ReactJS and other tech stack mentioned above, we have built a front end of survey builder as well as front end of the transcript generator. This will simplify the user to create a good survey and look at the responses and use the feature of transcript generator.

**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 using 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 of Survey Payload:**

{ "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 Backend:**

* 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*