Technical Description of the Website: http://qxviz.ai/

Yuqing Yang[‡†], Rasha Ali[⋆], Tripat Kaur[⋆], Boris Joukovsky[‡†], Nikos Deligiannis[‡†]
[‡]*Department of Electronics and Informatics, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium*
[⋆]*Faculty of Engineering, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium*
[†]*imec, Kapeldreef 75, B-3001 Leuven, Belgium*
Email: yuqing.yang@vub.be, Rasha.Ali.A.Ali@vub.be, tripatk1994@gmail.com, bjoukovs@etrovub.be, ndeligia@etrovub.be

***Abstract*—This paper describes the implementation of a website that performs a subjective evaluation of explainable artificial intelligence algorithms applied on the task of detecting artificially-generated images. The website development has been divided into two sub-systems: the back-end and the frontend. The frontend covers the part of the system which is visible to the user, i.e., it deals with the client side. Anything happening on the user side of the connection can be received or manipulated by the user. It deals mostly with the user interface and the user experience aspects of the website. The backend is another part of the system that deals with the core functioning of the website such as algorithm logic, database interactions, server architecture; that is, the back-end is an enabling sub-system for the frontend. The website can be accessed using the following url: http://qxviz.ai.**

I. Introduction

The three-tier architecture is the most popular implementation of a multi-tier architecture and consists of a single presentation tier, a logic tier, and a data tier [1]. For the development of the website, a similar architecture was used. The web layer (i.e., the presentation layer) is what is visible to the users in their browsers. At the presentation layer, the user interacts with the application. Accordingly, the logic of the application is located at the application layer. It is commonly referred to as the backend. It acts as the middleman between the application layer and the database layer. The database layer stores the entire data linked to the application (see Fig. 1).

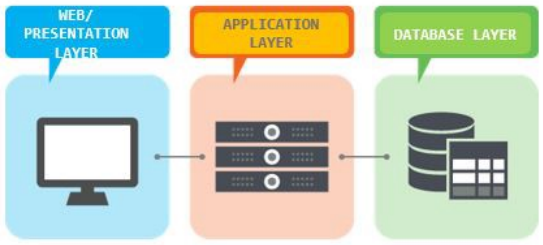

Fig. 1. The three-tier Architecture.

The frontend (i.e., the presentation layer) is developed using React JS, which is a modern Javascript framework for developing Single Page Web Applications. The back-end part (i.e., the application layer) is a REST API (also known as RESTful API), which is an application programming interface (API or web API) that conforms to the constraints of the REST architectural style. A REST API uses a GET request to retrieve a record in the Database layer through the application logic layer, a POST request to create one, a PUT request to update a record, and a DELETE request to delete one. All HTTP methods are used in the API calls to communicate with the frontend by exchanging JavaScript Object Notation (JSON), which is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa). These API calls are designed based on the logic needed by the explainable artificial intelligence algorithm applied on the task of detecting artificially-generated images.

II. Main Body

In the following section, the three modules of the architecture i.e. Presentation Layer, Application Layer and Database Layer are explained in detail.

A. Frontend

The frontend of the website is developed using React JS, which is a free and open-source frontend JavaScript library for building user interfaces based on UI components [2]. It is maintained by Meta and a community of individual developers and companies. The frontend has been designed using the React-Bootstrap library [3], which allows for easy development of UI components. The website has been deployed on http://qxviz.ai/.

The first page is the Home Page (see Fig. 2) that gives a basic description of the survey and some extra information about what a Deep Fake image is. The user sees a ”Get Started” button for signing up or signing in depending on whether the user already exists in the database.

A new user must fill in the Sign-In page, which requires the first name, last name, age, email address, and password to be filled in. Particularly, we collect the age of the user in order to avoid (after preprocessing) the bias brought by some user groups. After that the user logs in (in order to remember the user, his/her identity is stored in the form of cookies in the browser) and the survey begins. Meanwhile, the user details

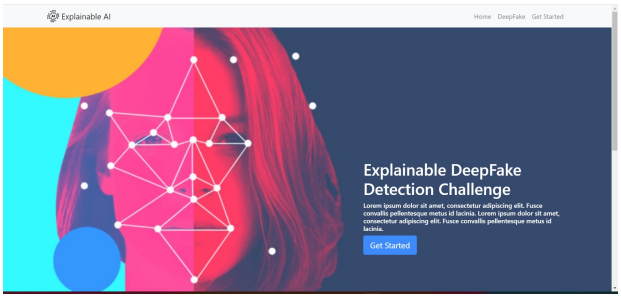

Fig. 2. Screenshot of the home page.

are sent to the backend survey in the form of JSON requests, the details of which can be found in the following section. Separate UI components have been built for the different types of questions that come as responses from the back-end.

The first question that the user receives is related to his/her background knowledge (see Fig. 3), where we want to identify the existing knowledge level of the user in the field of deep fakes and artificial intelligence(as this might have an impact on our analysis of the survey data). It is a single-answer question where the user can click on one of the available options.

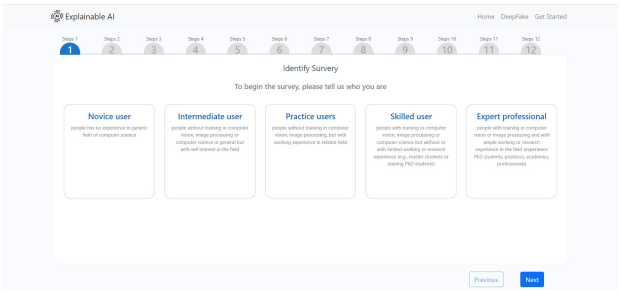

Fig. 3. Screenshot of the webpage with the background knowledge question to the user.

In the next screen the user is presented with an image and he/she is asked to visually examine the image and indicate whether he/she finds it fake or real (see Fig. 4). The user can zoom-in on the image to see its details more clearly. This functionality has been implemented using the react-image-magnify library[1]. Based on the response of the user, the next questions from the backend are made available to the frontend using HTTP requests.

In particular, it is important to state that the user's choice at this step determines the page that appears next on the website. There are four cases designed according to the actual label (a.k.a., real or fake) of the image and the user's choice:

- **Case 1:** The image is real and the user selects correctly. In this case, the user has correctly identified the actual label of the image, then five images are displayed on the screen (see Fig. 5), namely, the image, and four heatmaps produced by four different visual explainability techniques. These four techniques are self-attention [4], Grad-CAM [5], LRP [6] and Transformer attribution [7]. The user has to choose one of the heatmaps that they feel gives the best explanation (in the sense that the explanation attributes high importance to areas of the image that the user also find important in rendering an image fake).

[1]https://www.npmjs.com/package/react-image-magnify

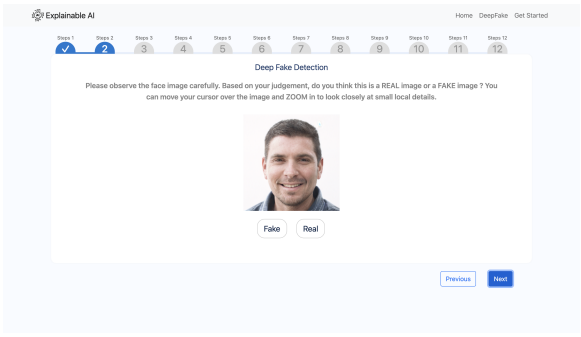

Fig. 4. The user is asked to visually examine the image and indicate whether he/she finds it fake or real.

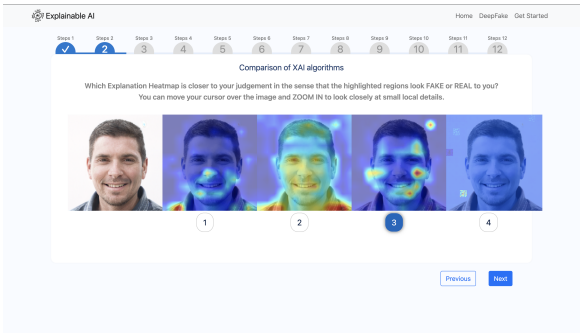

Fig. 5. The user is shown five images, namely, the image, and four heat maps produced by 4 different visual explainability techniques.

- **Case 2:** The image property is real, but the user selection indicates it is fake. In this case, first of all, the user is informed of his/her misjudgement (see Fig. 6). Afterwards, the user is shown the four heatmaps (similar to the ones before) and one more option, namely, *"I was not able to get any help from these heatmaps"* (see Fig. 7). As opposed to the case where the user chooses correctly, this step is an indirect evaluation that aims to explore whether users can learn from heatmaps and which heatmap is more stimulating to the user's perception.
- **Case 3:** The image is fake but the user selects it is real. As in Case 2, the user is notified of his/her mistake and then indicates whether he/she can draw useful information from the heatmaps. This is logically the same as in the second case.
- **Case 4:** he image is fake and the user correctly identified that. In this case, the user is requested to indicate the part of the image he/she based his/her judgment on; this is done by annotating parts of the image using rectangles. Again the user has the flexibility to zoom in on the image to see the details more clearly (this is also implemented

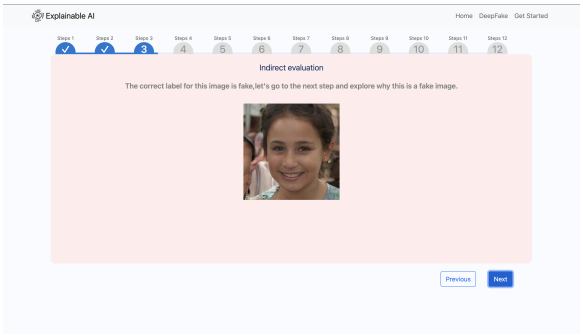

Fig. 6. The user is informed about the actual label of the image.

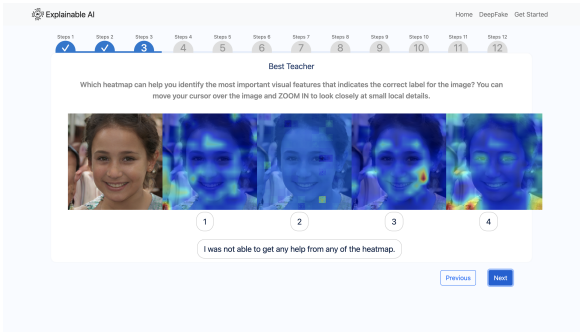

Fig. 7. The user is shown five images, namely, the image, and four heat maps produced by 4 different visual explainability techniques and one more option.

using the react-image-annotation library). The annotated image is sent to the backend using base64 encoding so that it can be saved on the server for later analysis (see Fig. 8). Afterwards, the screen shows five images, similar to the first case (see Fig. 5).

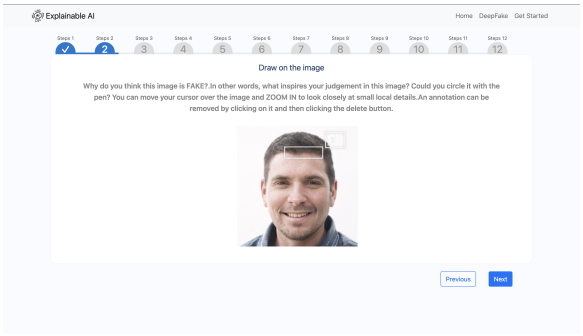

Fig. 8. By drawing rectangles, the user can indicate the part(s) of the image that drive his/her decision.

In addition, after the user has selected the heatmap generated by one of the techniques in any case, the next question contains two heatmaps generated by the selected technique as option (see Fig. 9). Although both heatmaps are from the same visual explanation technique, the two images are different in that one is from FaceImages (this dataset contains resized images from Flickr-Faces-HQ Dataset and fake images generated by StyleGAN [8]), and the other is processed using bilateral filtering. The purpose of this test is to assess and analyze whether, after applying bilateral filtering on the image, the accuracy of the heatmaps improves or not [9].

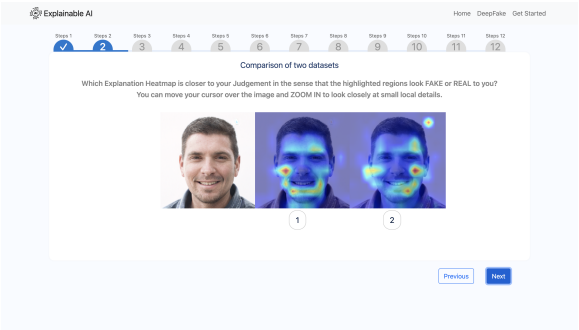

Fig. 9. The user is shown three images, namely, the image, and two heatmaps correspoding to an explanation for an image with and without applying billateral filtering.

This user is asked to perform the aforementioned evaluation for ten different images and their visual explanations. When they do an evaluation of the first image, a short video explaining what heat-maps are is shown to the users since this will help the users in answering the next questions. Moreover, the user needs to evaluate at least four images whereas the evaluation for the rest of the six images is optional. The user can decide to quit the survey using the "Exit Button" at the bottom left part of the screen. Just before exiting the survey, the user is asked to indicate his/her confidence while answering the questions (see Fig. 11).

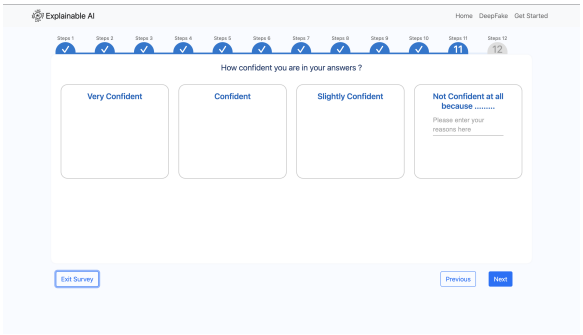

Fig. 11. The user indicates his/her level of confidence in their answers.

For all questions, a basic validation rule has been applied in that the user cannot go ahead without answering the current question. If the user tries, he/she will receive a notification message. Furthermore, the user is given the choice to go back to the previous question, review and potentially changing his/her answer. The user can view his/her progress with the survey at the top of the screen, a.k.a.,the completion timeline. Every time the user moves ahead a part of the completion timeline gets progressively filled.

For the styling of the user interface (UI), Sassy Cascading Style Sheets (SCSS) have been used. This facilitates us to write clean, easy and less Cascading Style Sheets (CSS) in a program construct. It contains fewer codes so you can write CSS quicker.

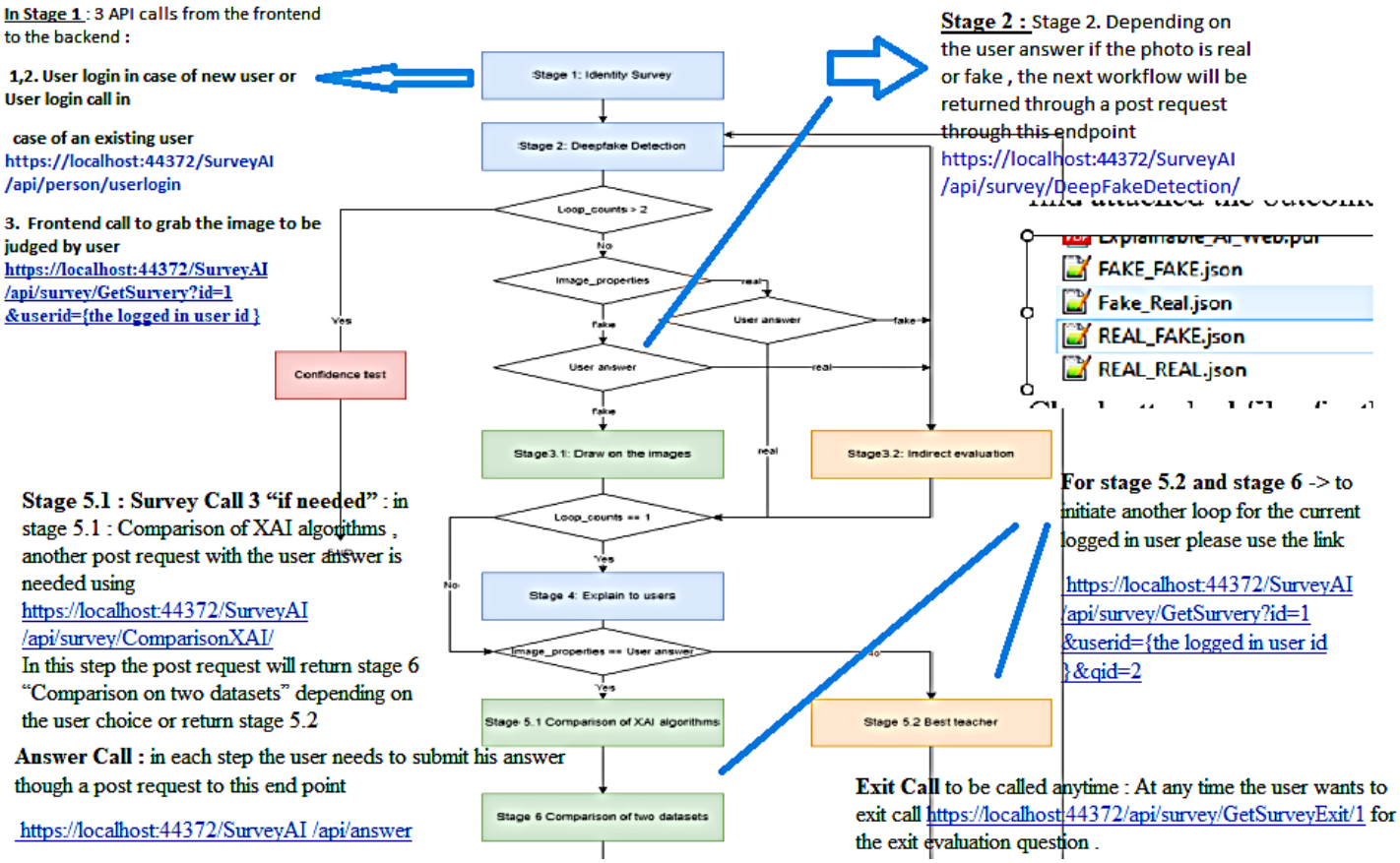

Fig. 10. API Calls against the survey flow chart.

B. Back-End

The back-end of the website has been developed using the C# language following the Models-Views-Controllers (MVC).NET framework architecture with Microsoft Visual Studio 2019 [10] which is an integrated development environment from Microsoft. The communication between the frontend and the back-end is based on Hypertext Transfer Protocol (HTTP) requests, made by the client (frontend) to a named host (back-end) located at a server. The back-end layer is hosted on the IIS (Internet Information Services) server, which is an extensible web server software created by Microsoft for use with the Windows NT family. The back-end is divided into three main endpoints that are called controllers. They are the `Person Controller`, the `Answer Controller`, and the `Survey Controller`. Each endpoint (a.k.a., controller) is consumed by the frontend to execute the required functionalities of its designated role.

Based on the survey flow cart, we present a detailed description of the corresponding API calls (see Fig. 10):

- The first step in the website cycle is the user sign-in or registration, which are based on HTTP get/post calls from the frontend. To begin with, the frontend sends the user details in the form of JSON objects, containing the username and password (see Fig. 10). Upon registration, the frontend initiates another HTTP call to the back-end which will response back by a JSON object that contains the first two steps of the survey (called step 1 and step 2 of the survey steps: user login in case of new users or user login call in case of an existing user).
- Then, the user is asked to visually examine the image, and there is a frontend call to grab the image under examination. The infromation that has been sent contains the image. When the user clicks next the frontend sends back the user's response to the back-end and initiates an HTTP call to the API. The response returned back by the previous call from the back-end depends on the answer of the user and the label of the image (the image is fake or real and the user selection is fake or real). That leads us to have these 4 combinations mentioned in Section II-A. For each combination, the back-end forms the correct response and sends it back.
- Finally, upon finishing up the survey, the user is asked to answer one final question, i.e., to indicate its confidence. To this end, the frontend initiates a call to the end point GetSurveyExit in SurveyController that returns back the exit question to the user, whose answer is saved in the

database.

In summary, throughout each step of the survey, when the user clicks on next button, the frontend initiates an http post call to register the user answer and wraps it in a JSON object using the http answer submit call. The API calls between the back-end and frontend are summarized in Fig. 12.

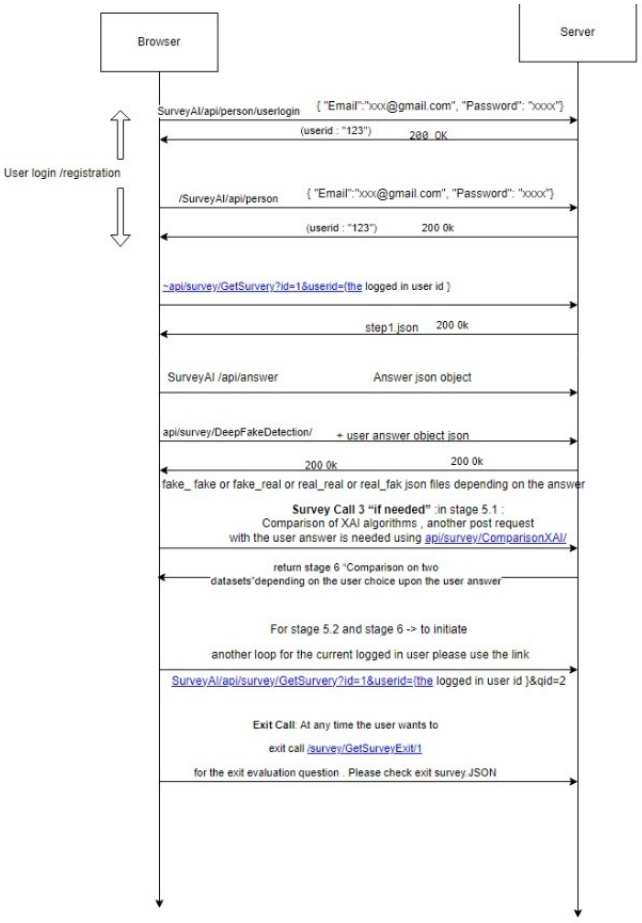

Fig. 12. Back-end API HTTP Calls.

C. Database

The database layer stores the entire data linked to the back-end. The database contains not only information about our the image and explanations datasets and the content of the survey questions, but also information about user registrations and the results of the survey.

The database is designed using MySQL [11], which is an open-source relational database management system. "SQL" is the abbreviation for Structured Query Language that is used to query the database; see Fig. 13.

D. QXVIZ Deployment

After setting up the three-tier architecture, in order to deploy the website, the command `npm run build` should be executed on the local machine. This creates a build folder

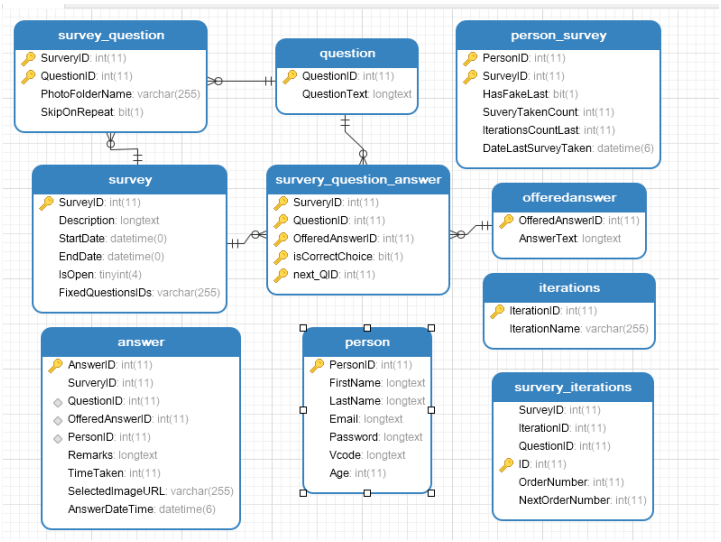

Fig. 13. The Database ER Diagram.

in the project directory, which can then be moved to the server machine.

For the back-end, the entire solution (code) needs to be built, compiled and then published in a folder. In Visual Studio 2019, you can right-click on the *project* (not the solution) in the Solution Explorer and select *Publish* in the Publish tab. Publishing creates the set of files that are needed to run the application. In order to deploy these files, they must be copied to the target machine and placed into the publish folder that must be set up under the Internet Information Services (IIS) server. Moreover, the IIS server must be installed on the server machine running windows operating system. IIS is a flexible, general-purpose web server from Microsoft that runs on Windows systems to serve requested HTML pages. An IIS web server accepts requests from a remote client (here in our case the frontend) and returns the appropriate response.

For the database, MYSQL Community Edition is required. It is a free downloadable version, maintained by an active community of open source developers. MySQL Cluster Community Edition is available as a separate download. You can see the SQL transcript file for the whole database tables is attached as appendix, which can be used to create the needed tables and database for our website. Moreover, to access the MYSQL database, we need to install DBEAVER which is an SQL client software application and a database administration tool. It uses the JDBC application programming interface to interact with relational databases via a JDBC driver. For other databases, it uses proprietary database drivers.

REFERENCES

[1] E. F. (n.d.). Whitepapers. Amazon., "Three tier web application," https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html Retrieved September 28, 2022.

[2] W. Foundation., "React-wikipedia," https://en.wikipedia.org/wiki/React_(JavaScript_library Retrieved September 28, 2022.

[3] B. react. (n.d.)., "React-bootstrap," https://react-bootstrap.github.io/ Retrieved September 28, 2022.
[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.
[6] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: an overview," *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 193–209, 2019.
[7] H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 782–791.
[8] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.
[9] H. Wang, X. Wu, Z. Huang, and E. P. Xing, "High-frequency component helps explain the generalization of convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8684–8694.
[10] V. S. Community, "Visual studio community," https://visualstudio.microsoft.com/vs/community/ Retrieved September 28, 2022.
[11] S. all popular databases: MySQL, "Universal database tool," https://dbeaver.io/ Retrieved September 28, 2022.