ProcSmart

Software Requirement Specification

Object Oriented Design (CSE - 687)
Syracuse University
New York
October 2020

Table Of Contents

1 Introduction

1.1 Purpose

The purpose of the document is to provide a comprehensive description of ProcSmart, a smart online proctoring system for real time activity monitoring of candidates during an online examination. This document will explain the system's purpose and services as well as the constraints under which the system must operate. This SRS is written for system stakeholders including users, developers, testers, project managers and software architects.

1.2 Product Scope

Although we already have a few online proctoring systems in place, now due to the pandemic, the need of a smart online proctoring system is extensive and obviously the existing ones are not efficient enough to gain our complete trust. The answer is quite simple and straight forward. The ongoing public health crisis has prompted colleges and organizations to go online for tests and remote proctoring. Be it a university exam or an onsite interview with an organization, everything is going virtual. And it's human tendency that given a chance, we start trying to compromise the system. That's definitely against the integrity policy everyone must follow.

The application intends to solve the problems faced by existing proctoring systems and include some extra features, which will enable better efficiency. The issues which persist in the current proctoring system are as follows:

- In the live proctoring, a professional and active proctor keeps track of the session by the help of a webcam. Human interference is required here.
- Also, taking sessions of students by grouping them costs a lot of time and resources.
- With the human intervention the system is not completely bias free.
- In recording proctoring, the whole session gets recorded and is later viewed by an instructor, which again requires human intervention.
- Also, storing the recordings requires a huge chunk of memory.

Why is it a killer app?

Our idea deals with an autonomous proctoring system in which there won't be any live proctoring or recordings, rather the system will monitor the session. The most advanced form of autonomous proctoring is to develop a smart system which can perform proctoring on its own. Similar to previous types of proctoring, student's audio-video and screen sharing feeds are recorded. But apart from that, a smart proctoring software, that uses video and audio analytics tries to find suspicious activities. This technique overcomes the drawback of the previous two methods. It does not require any human review and therefore is extensible, cost effective and bias free. There is no need to store recorded sessions, instead the system generated logs will be stored. In case of any suspicious activities, the system itself will flag a user. It also authenticates the student taking the test and avoids any impersonation. The users need to grant access to their microphone and webcam. The app also takes care of and respects user privacy and security.

1.3 Definitions, Acronyms and Abbreviations

1. *Administrator* - This is the system Admin who has special rights. These special rights include being able to view logs, create session and flag users.
2. *User* - This role is defined for students who will use the application to give exams.
3. *Flag* - This represents the Flag that can be set on users if they violate examination constraints. This is used as a method to identify users whose session had anomalies.
4. *Session* - This refers to an exam session that is scheduled by the Administrator. Students join these sessions to give their exams.
5. *AI Models* - These are the Artificial intelligence models developed to perform tasks like facial recognition, unauthorized object detection and so on.

2 Overall Description

2.1 Product Perspective and Functions

1. Administrator will login using the credentials and if validated correctly, he/she will redirect to the administrator dashboard. Then the administrator will select the students who are eligible for a particular exam. Administrator will make the exam available for the students.
2. User will login using the credentials and if validated correctly, it will redirect to the user dashboard. User can see the exam session once it is available. User will start the exam and give permission to access microphone and webcam. In case user denies permission, it will automatically lead to user dashboard and user will need to restart the session.
3. The application will use facial recognition, object detection, spoofing for actively monitoring students taking an exam. It will generate logs based on monitoring and activity tracking and the logs will be uploaded simultaneously to the database. In case of any suspicious activities, the system will flag the user.
4. The admin dashboard will be updated in real time with notifications from the generated logs. In case of any discrepancies, the admin can override the flags set by the system.
5. The application will close automatically at the end of the session. User has no direct interaction with the application. Once the user submits the exam or if the exam gets auto submitted due to the time constraint, user does not have to explicitly close the application, rather it will close automatically.

2.2 Design and Implementation Constraints

1. One of the critical aspects of our application is its inherent dependency on a stable internet connection. Since the app runs on a client-server architecture, it is imperative to have an active internet connection.

2. In addition to a stable internet connection, it is also necessary to have a camera and microphone in your computer as proctoring is done using both real-time audio and video.

3. We have used *YOLOv3* to perform object recognition in our AI models. Thus, the system is dependent on the API being available always. In addition, the accuracy of the Models would depend on the latest version of this API.

4. The quality of the camera being used also has a lot to do with the accuracy of the models, we are reliant on the video being decently clear for our models to perform at their best.

2.3 Assumptions and Dependencies

2.3.1 Assumptions

1. The system will be deployed on a WSGI server, thus we make an assumption that it is available without interruptions.

2. The system also heavily relies on the AI models being used. An assumption that the future version of the libraries does not drastically change the algorithms is made.

3. The system will not be able to detect if some object is placed in front of the camera blocking the view of the face, we assume that the user's face is initially visible whenever they join the session.

2.3.2 Dependencies

1. The system will rely on the IDE (VS Code) and database tools to manage and maintain the system.

2. The system will depend on the server we choose to host it.

3. The system rely heavily on the python libraries *OpenCV, YOLOv3, and Tensorflow* for the performance of the AI models.

4. The performance of the system will heavily depend on the processing power of the server it is hosted on, as the AI models need good processing power to work at their best and fastest.

3 Interface Requirements

In this section, we will discuss about the user Interface and software interface requirements.

3.1 User Interfaces

We have two actors in the application - Admin and User. Both the users are managed by the same interface, which is developed using HTML, CSS, JavaScript and Bootstrap framework. At the time of login, system will decide whether the actor is an admin or a user.
The app must be accessible on different devices which has an internet connection and a modern browser.
The first page of the application is the login page. Service layer will verify the credentials of the user as well as the authorization level. Based on the authorization level, user will reach the dashboard page.

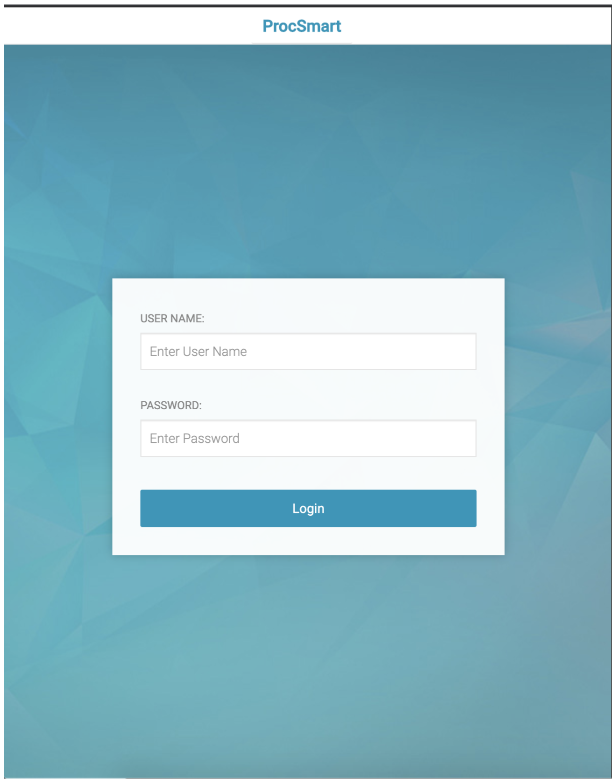

Figure I: Login View

Since ProcSmart is mainly used by the admin, lets discuss the options an Admin will have on the dashboard screen. The main feature the admin will have is to check the student's logs after the test. Admin will be provided with a search box where he/she can search a particular student and check the logs. If admin selects a student to check the logs, a http json request will be sent to RESTfull API with the student id, service will fetch the data from the database and list the logs on the screen.

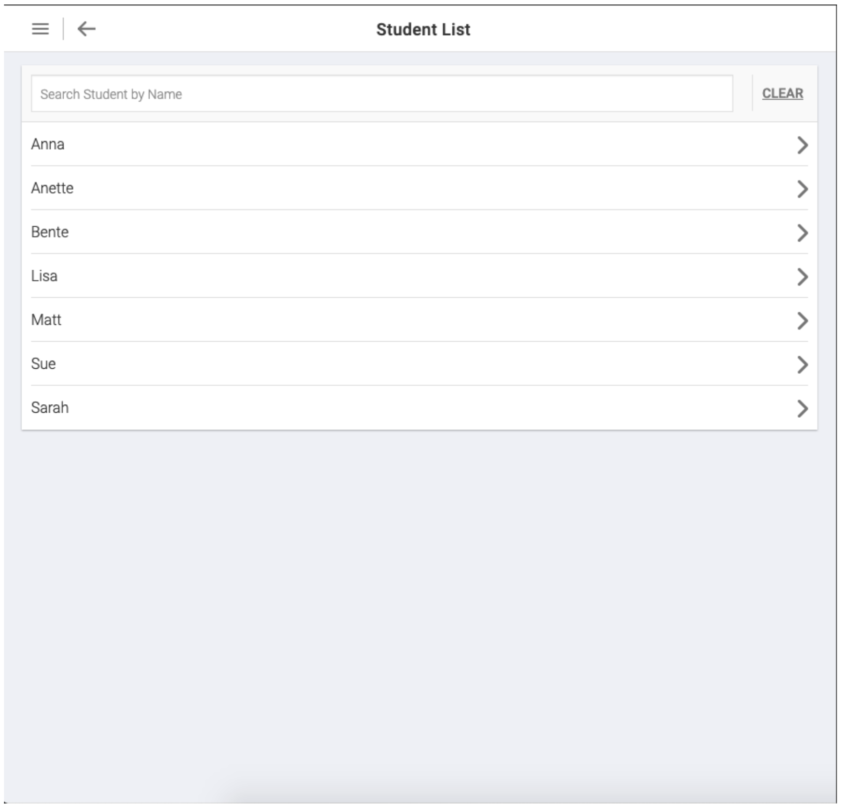

Figure II: Student View

Admin will also be provided with an interface where he/she will be able to create/delete new users(Admin or Student). An http request will save/delete the data in/from the database via the service layer. There will be a separate page which will enable admin to create a test session and add different students for that particular session. Once a student is enrolled in a test session they will see the sessions in their dashboard.

A Student has quite a few options to interact with the application. However, triggers the most important block in the application. After successful login, the student will be taken to dashboard page where he/she could see the list of exams session he/she has been enrolled in. A student can start the session by clicking on the start session button. Once the session start request is sent to service, the service will now initialize the AI models for that particular session and student combination, after which AI model starts returning logs to service. Service layer now has the responsibility to process those logs and save them in database. After completing the test or any time in between, the student will have the option to end the session.

3.2 Software Interfaces

Our system will use YOLOv3, openCV, Tensorflow, audio recognition libraries and a custom RESTful Service to implement all of the functionality.

- *YOLOv3* - This is the latest variant of a popular object detection algorithm YOLO –

You Only Look Once. The model recognizes 80 different objects in images and videos, but most importantly it is super fast and nearly as accurate as Single Shot MultiBox. We use this algorithm to recognize objects like notebooks and mobile phone in the frame. As we use this model to recognize objects, the accuracy will be exactly the same.

- *OpenCV* - This is an open source computer library that is used for machine learning and computer vision. This was build as a common infrastructure for all computer vision applications. We use this to perform all tasks related to computer vision. Even the fore mentioned model YOLOv3 works in conjunction with OpenCV. This library will be used in every AI model to help out with machine learning.
- *Custom REST API* - We will create a custom REST API for the application to communicate with. This service will then dump all the data into the database. It will also be responsible for fetching all the data like logs, list of users for the Administrator dashboard. We will expose certain endpoints which the application will use to communicate with it.
- *Tensorflow* - We will also use tensorflow to help the machine learning algorithms perform faster. We also make use of certain models from the tensorflow library.
- *Audio Recognition* - To perform audio protoring we use three python API's : speech_recognition, nltk, pyaudio.

4 Requirements

4.1 Functional User Requirements

UR.1 Each user has their own private account. If a user forgets his/her account or password, he/she should be able to retrieve it. Also, they should be able to change their password.

UR.2 Each administrator should have the permission to log in to a web page, which provides them with a dashboard to manage users and logs.

UR.3 Users / Students should be able to start the session after logging in and granting permission to microphone and camera.

UR.4 Administrators should be able to add or remove users and change the password for student users.

UR.5 Administrators should be able to view logs for each of the sessions that the students took.

UR.6 Administrators s have the right to flag/report the student in case of any observed anomaly.

4.2 Functional System Requirements

SR.1 The app will make use of a custom login service to login into the system.

SR.2 The database holding the user's information should be available whenever the user wants to login.

SR.3 The database holding the user's information should be updated with the new password as soon as the user decides to change.

SR.4 Administrators should have special grants/rights than other users. The database should be able to serve up logs on demand to the administrator.

SR.5 The system should be able to differentiate between student users and administrators and accordingly adjust what to display.

SR.6 The system should provide AI models for :-

- ***Face Recognition*** – Used to detect various features like mouth and eyes and track movement.
- ***Object Detection*** – Identify objects like mobile phones, notebooks etc.
- ***Spoofing*** – Used to detect spoofing using photos or static prerecorded video

SR.7 The application should allow the administrator to view all session logs for students.

SR.8 A database will be modeled for storing all real-time data.

SR.9 The service layer will act as an intermediary between the AI model and the Database

SR.10 The system should be able to generate meaningful logs based on the output of the AI model.

4.3 Non-Functional User Requirements

UR.1 The app should be able to be used on any browser on mobile or laptop with an internet connection.

UR.2 Users personal information and the video being monitored should be stored privately and securely

UR.3 Users should not be logged into the application on multiple windows, concurrent logins should not be allowed.

UR.4 The user interface should be user friendly and users should be able to navigate through the app with ease and with minimal training.

UR.5 The logs created should be updated into the database immediately so that they can be reflected on the administrator's dashboard in real time.

UR.6 The app should be accessible only to authorized users.

UR.7 Users should be able to restart the session in case of disconnection due to internet issues.

4.4 Non-Functional System Requirements

SR.1 System should allow users, who are using a browser, to access the app.

SR.2 System should allow this app to operate on all range of devices as long as it has a web browser

SR.3 System should keep the users' information and the video being monitored secure and safe

SR.4 The models should run fast with minimal lag in the processing.

SR.5 The models should not require a lot of processing power or memory.

SR.6 The system should provide an easy to understand and intuitive interface to the user.

SR.7 The system should ensure integrity of the data and should ensure that the data is updated synchronously.

SR.8 The system should monitor the web bandwidth and adjust its processing technique accordingly.

SR.9 The system should have a way to retrieve data or access a lost account.

SR.10 The system should have good software attributes such as maintainability, dependability, efficiency and acceptability.

SR.11 The system should provide the AI models/algorithms on demand when a session is started by a student.

5 Models

5.1 Activity Diagrams

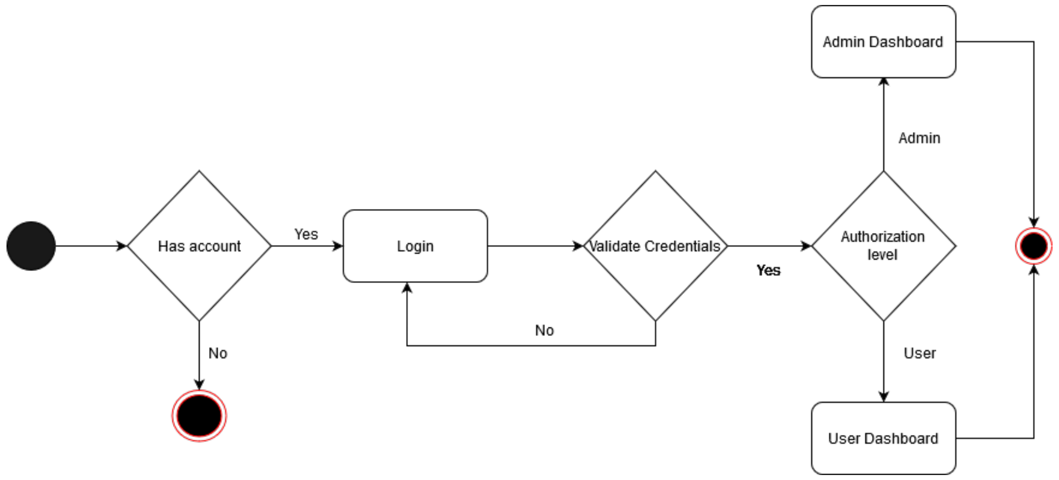

Figure III: Login Activity Diagram

- The activity diagram for login represents how a user's credentials are validated with the information available in the database.
- There are two kinds of users, administrators, and normal users. Administrators are the proctors, and the users are students who will take the examinations.
- A few initial administrators are created by the developers and these administrators can generate users (students) and their passwords for the examination.
- Once the credentials are entered to login, the system determines the type of user and access is provided accordingly. Students or normal users can login using the credentials generated by the administrator, but these accounts have limited functionality.
- If the credentials provided by the user are incorrect, the validation fails, and the systems throws an error that requests the user to retry.

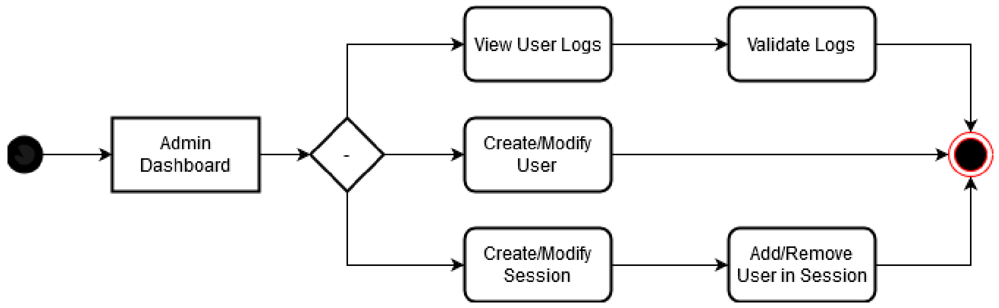

Figure IV: Admin Activity Diagram

- The activity diagram for administrator represents the flow of activity for an admin once he/she logs into their account.
- As shown in the diagram, the initial dashboard provides the admin with three main functionalities, i.e. to view user logs, create or modify existing user and create or modify existing test session.
- The admin can view user logs, these logs are available in real time and the administrator has the rights to validate the logs. He can either flag a student for academic integrity issues or unflag a false positive flag set by the artificial intelligence model.
- The admin can also create new users that he or she would later assign an examination to. The admin can also modify information of the existing users.
- The admin can create new test sessions and add or remove users from this specific test session. The admin can also modify an existing test session.

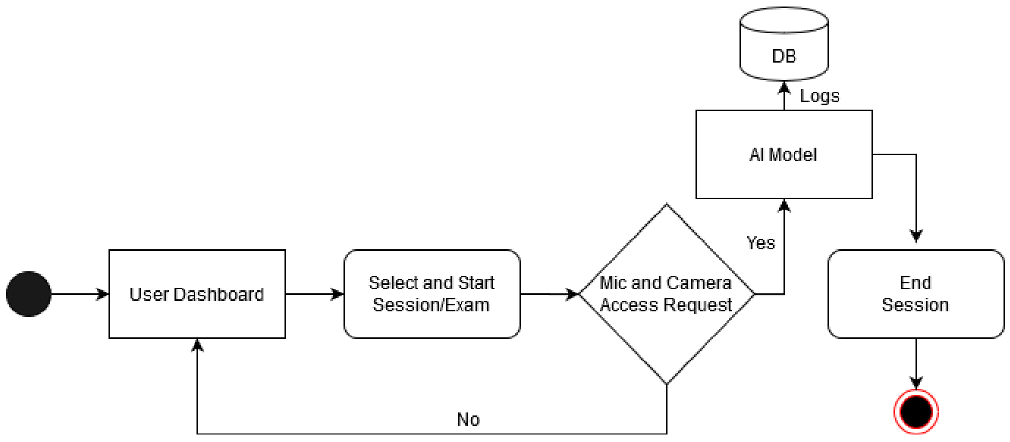

Figure V: User Activity Diagram

- The activity diagram for student(user) login represents the flow of activity for an user once he or she logs into their account.
- As shown in the diagram, a student(user) has limited functionality and they are allowed only to take an examination assigned to them.
- Once the user selects to start an exam session, he or she is requested for approval to access camera and microphone.
- If the request is denied, they would be taken back to their dashboard. If the request is approved, the AI models would run in the background and log their activity in the database.

5.2 Use Case Diagram

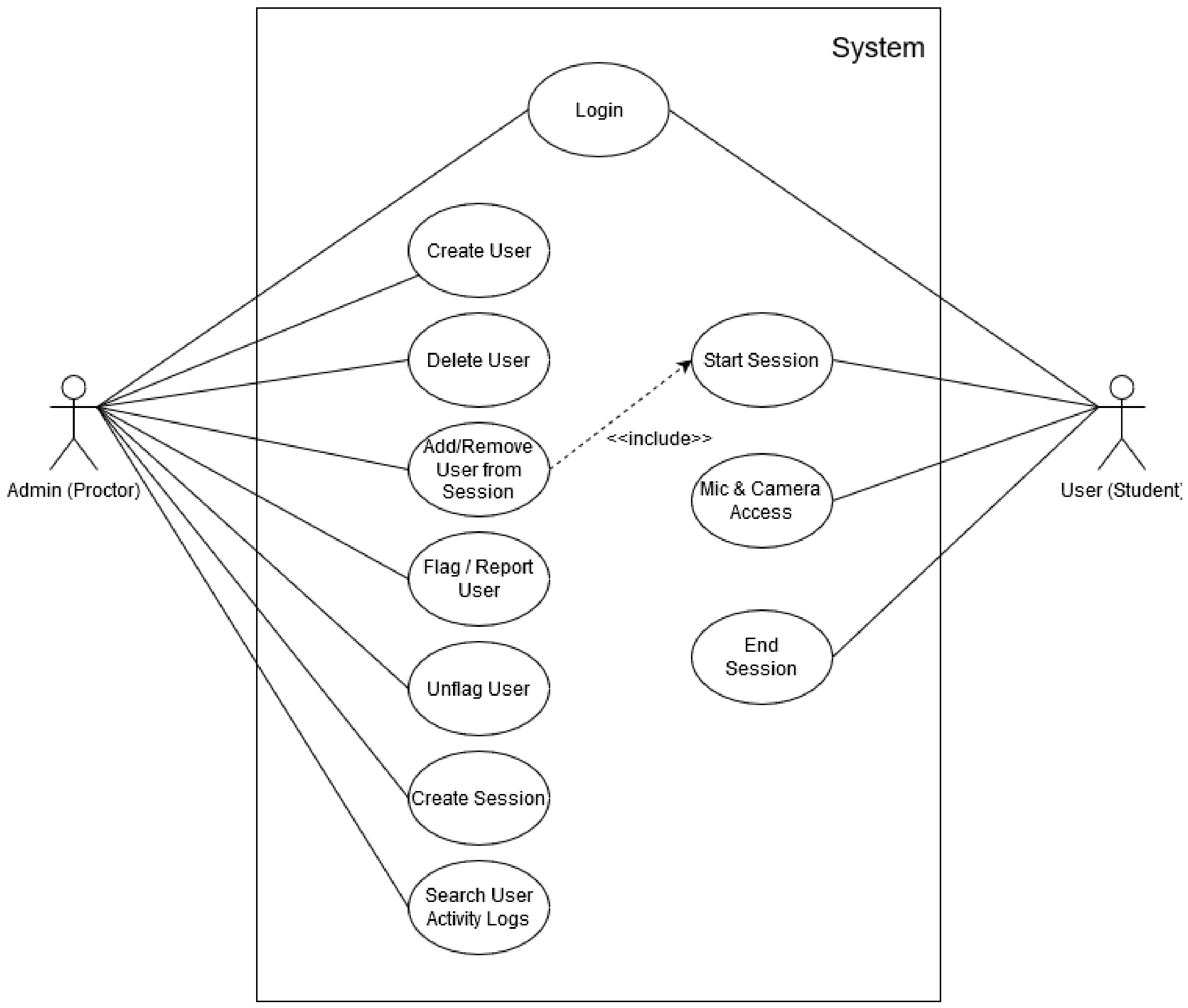

Figure VI: Use Case Diagram

- *Login* – The administrator and users can use the login functionality to login to their respective accounts.
- *Create User* – The administrator can use this functionality to create new user (student) account.
- *Delete User* – The administrator can use this functionality to delete an existing user (student) account.
- *Add/Remove User from session* – The administrator can use this functionality to add or remove an existing user from a test session.
- *Flag/ Report User*- The administrator can use this functionality to flag an user that the AI model logs as an issue of academic integrity.

- *Unflag User* – The administrator can use this functionality to unflag a false positive report in the log from the AI model.
- *Create Session* – The administrator can use this functionality to create a new test session.
- *Search User Activity log*- The administrator can use this functionality to search for a specific user in the logs generated by the AI model.
- *Start Session* – The student can use this functionality to start taking a test session that is assigned to them by the administrator.
- *Mic and Camera Access* – The student can use this functionality to approve or deny the request to access camera and microphone.
- *End Session* – The student can use this functionality to end an ongoing test session.

5.3 Sequence Diagrams

- To login to the application, once the user (student or admin) enters the credentials to login, the application verifies the credentials against the records in the database. If the credentials are correct, the user is logged into their dashboard.
- The admin can create new users, once the admin requests to create a new user, the application verifies if an user exists already with the same credentials, if user already exists, the system throws an error and if not, a new user is created.
- The admin can create new test session and assign them to existing users, this initiates the AI models which in turn start sending logs to the database about the new session.
- The search user logs functionality lets the admin search for a specific user and returns the logs for that user.
- The admin can also flag a user based on the logs reported by the AI model and unflag a user that the system flagged (false positive).
- The admin can use the logout functionality to logout of the system.
- The user can login to the system and credentials are verified similar to the admin credential verification but once logged in, the user can only start the test session assigned to them.
- Once the user decides to start a test session, they have to approve camera and microphone to begin the test and then end session to exit out of the examination.
- The user can also use logout to exit out of the system.

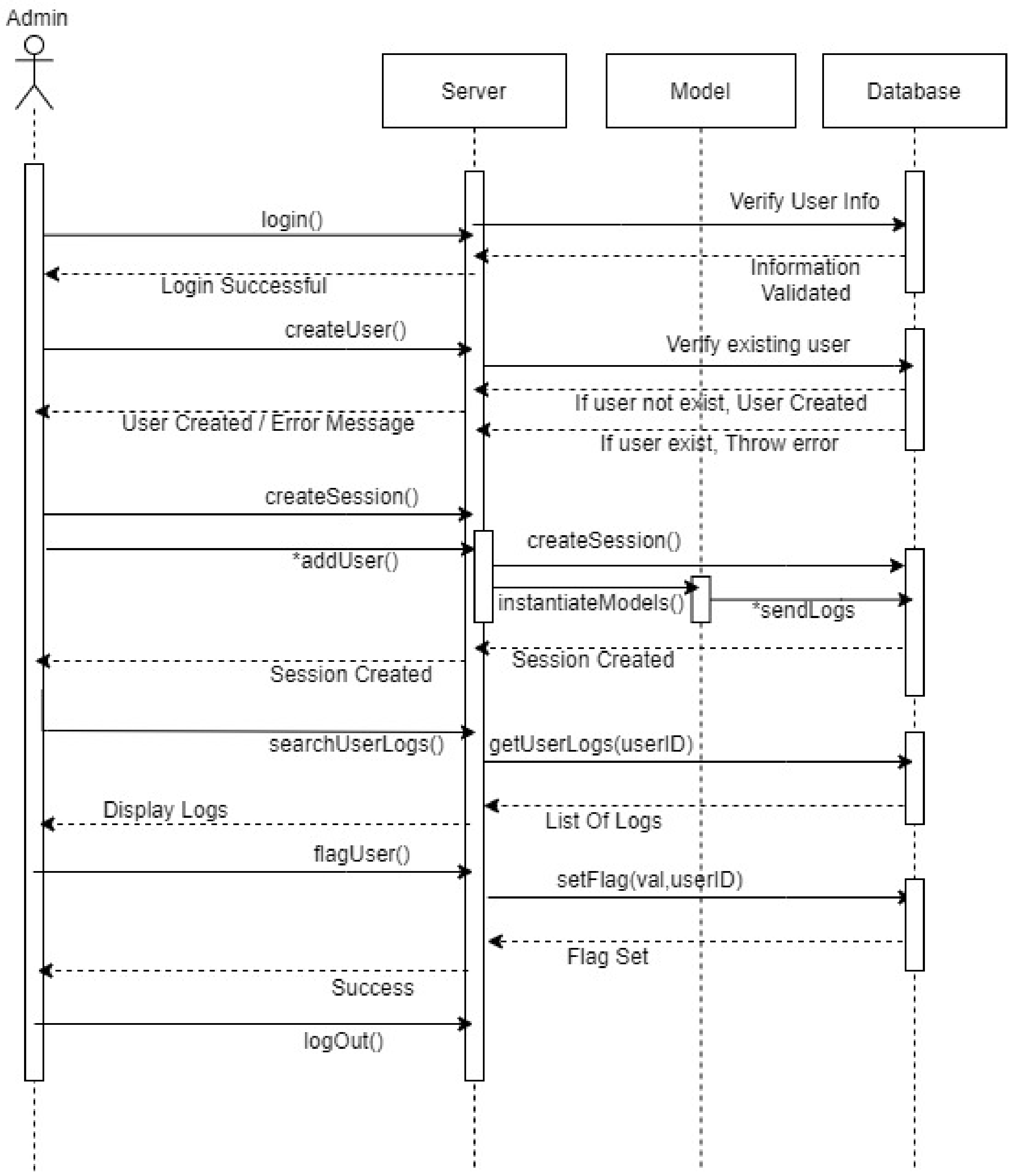

Figure VII: Sequence Diagram

5.4 Class Diagrams

- Admin and User class inherits All Users class.
- Admin is responsible for creating a new session. The admin can also delete an existing session. Once the session is created, the admin can add/remove users to/from the session.

- User can join a session which is assigned by the admin and once the exam is over he/she can exit the session. All the users will be assigned a default flag=false, which can be overridden by System/Admin.
- Once a session is created, a unique sessionId is assigned, which can be used to retrieve information about the session. When a user starts the session isActive flag is set to true.
- Logs are saved for each user and can be retrieved by admin. Once the logs are retrieved the admin can set or unset the flag value for the specific user.

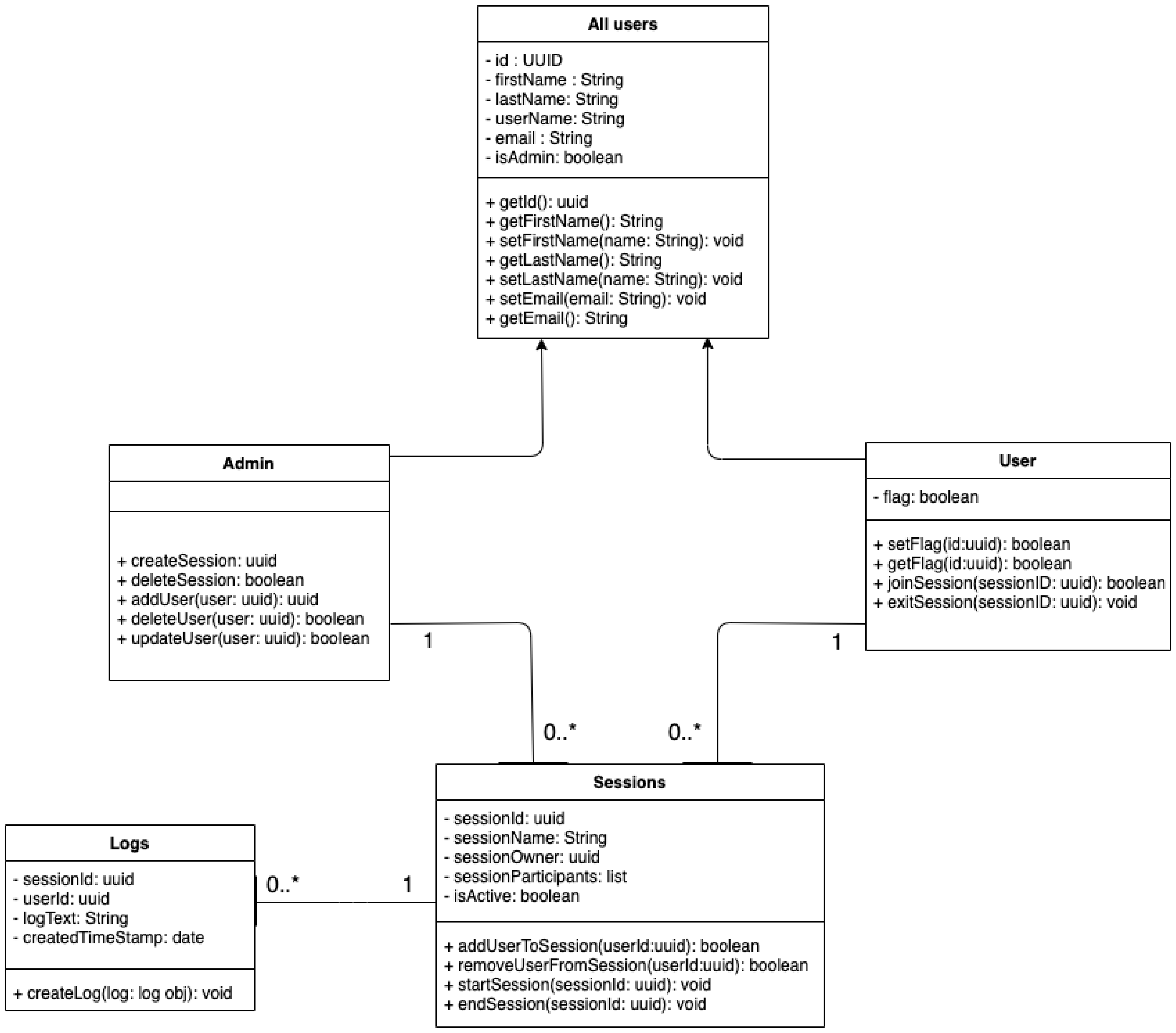
All users
- id : UUID
- firstName : String
- lastName: String
- userName: String
- email : String
- isAdmin: boolean
+ getId(): uuid
+ getFirstName(): String
+ setFirstName(name: String): void
+ getLastName(): String
+ setLastName(name: String): void
+ setEmail(email: String): void
+ getEmail(): String
Admin
+ createSession: uuid
+ deleteSession: boolean
+ addUser(user: uuid): uuid
+ deleteUser(user: uuid): boolean
+ updateUser(user: uuid): boolean
User
- flag: boolean
+ setFlag(id:uuid): boolean
+ getFlag(id:uuid): boolean
+ joinSession(sessionID: uuid): boolean
+ exitSession(sessionID: uuid): void
1
1
0..*
0..*
Sessions
- sessionId: uuid
- sessionName: String
- sessionOwner: uuid
- sessionParticipants: list
- isActive: boolean
+ addUserToSession(userId:uuid): boolean
+ removeUserFromSession(userId:uuid): boolean
+ startSession(sessionId: uuid): void
+ endSession(sessionId: uuid): void
Logs
- sessionId: uuid
- userId: uuid
- logText: String
- createdTimeStamp: date
+ createLog(log: log obj): void
0..*
1

Figure VIII: Class Diagram

5.5 State Diagram

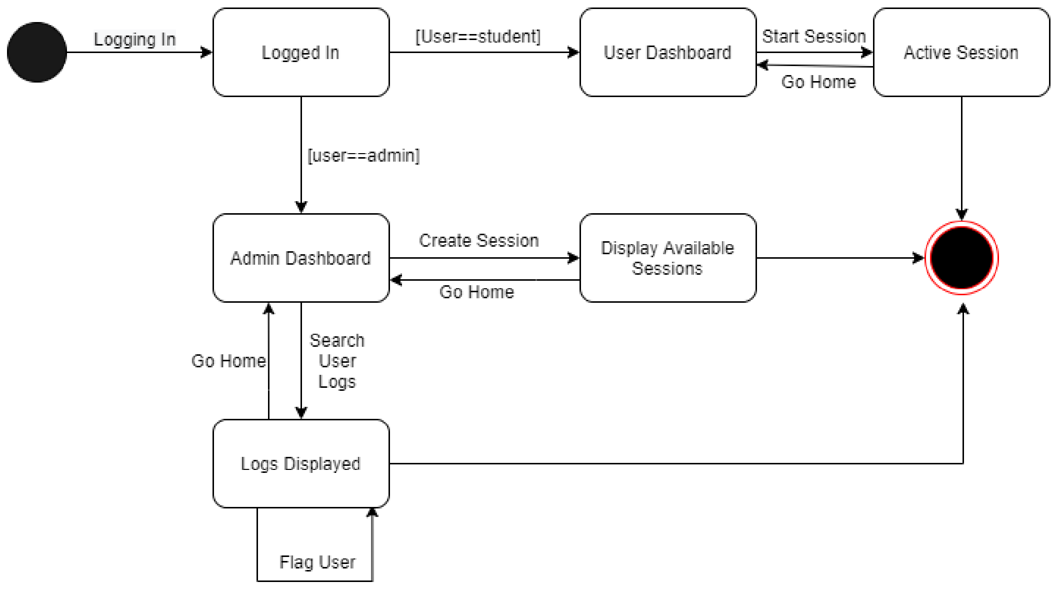

Figure IX: State Diagram

- The state diagram shows the state of users.
- The initial state of user is logged-in, from here the user is directed to either to the user dashboard or to the admin dashboard based on the authorization level.
- From the user dashboard, the user can start a session and end session. Once a session is started, they are in active session state.
- From the admin dashboard, the admin can create session and go back to dashboard. Once a session is created, the admin is in display available sessions state.
- The admin can also display user logs by searching for users. The admin will then be in logs displayed state and can chose to flag or unflag the user. The admin can also chose to return to their dashboard.

6 System Architecture

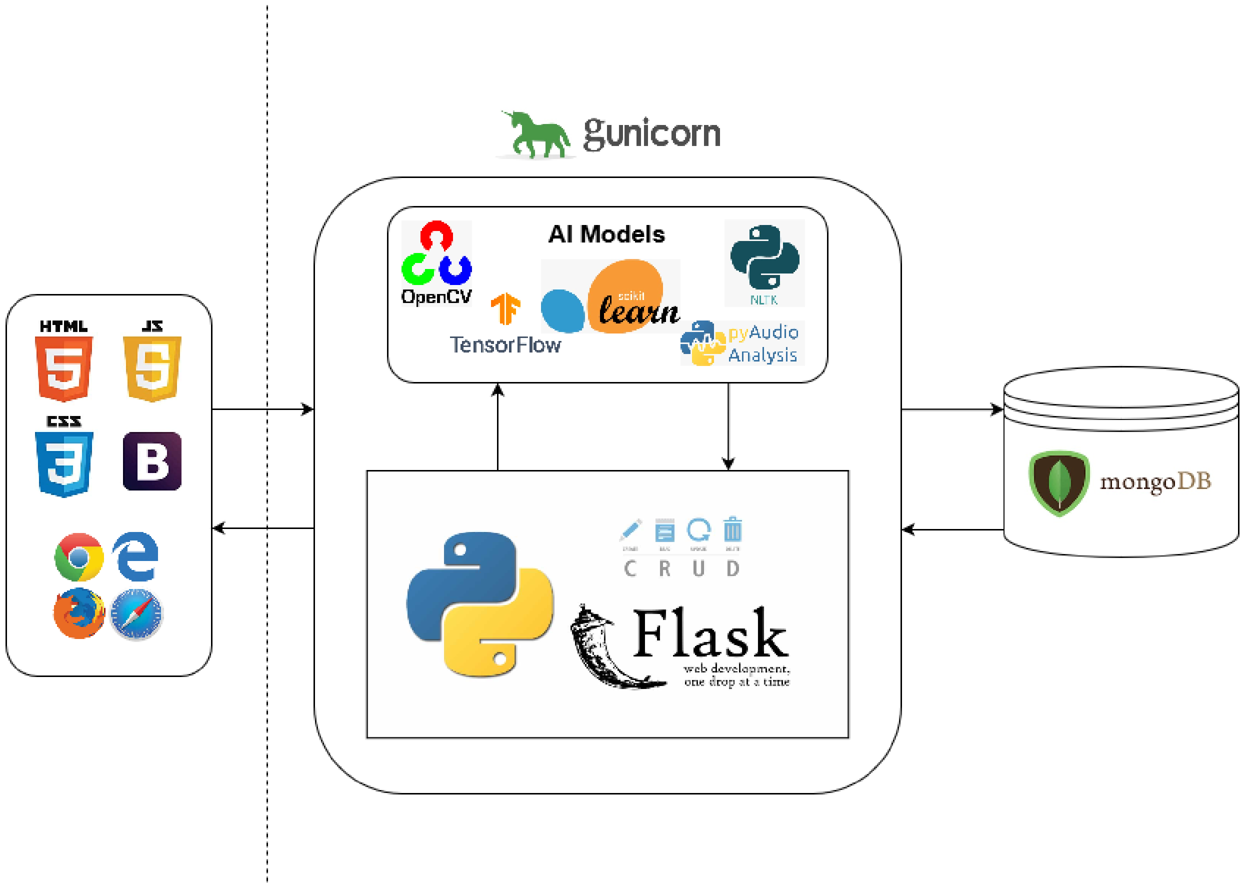

Figure X: System Architecture Diagram

6.1 Front-end

Front-end is developed using HTML, CSS, JS, Bootstrap. User will use Login Page and Dashboard pages to interact with the application. Since our application is not UI heavy we have tried to keep front-end.

6.2 Back-end

The service layer sits at the centre of our application is responsible for serving the user request and interacting with AI models. Below we have listed the about the web framework Flask and the WSGI server we are using in the application.

Python Flask

We are using Python flask, which is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

WSGI Server

For deploying our web application we are using Gunicorn (Green Unicorn) a WSGI HTTP Server. It is a pre-fork worker model. The Gunicorn server is broadly compatible with a number of web frameworks, simply implemented, light on server resources and fast.

6.3 Machine Learning Models

The core of our project are different machine learning models which analyses the test takers activity during the exam session and send the logs back to service which are parsed, transformed and eventually saved in the database. Below is the list of different Machine learning libraries and APIs

Tensorflow: TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries. In the project tensorflow is used for face landmarking and detecting unauthorized object during the test session.

OpenCV: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. We have used OpenCV to detecting mouth opening and eye movement.

face-recognition: It recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

Sklearn: Sklearn is the most widely used library for machine learning. We are using sklearn for implementing face spoofing model.

Bibliography

[1] Face Recognition API : `https://pypi.org/project/face-recognition/`.

[2] TensorFlow 2 Object Detection : `https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/`.

[3] Web service using Python Flask : `https://opensource.com/article/17/3/writing-web-service-using-python-flask`.

[4] Sound Pattern Recognition : `https://medium.com/@almeidneto/sound-pattern-recognition-with-python-9aff69edce5d`.

[5] YOLO object detection with OpenCV : `https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/`.

[6] Loading images with OpenCV : `https://pythonprogramming.net/loading-images-python-opencv-tutorial/`.