Software Design Description

for

Fake Attendance Signature Detection Application

Prepared by Alperen Kayhan, 2388532 - Ali Efe Sarıgöz, 2385664 - Bora Bedirhan Uyar, 2526788 - Halil Burak Özdemir, 2316107

Middle East Technical University NCC

Computer Engineering

Supervised by Meryem Erbilek

17.11.2025

Contents

Table of Figures

1 Introduction

1.1 Summary

The Fake Attendance Signature Detection Application (FASDA) is a solution that will facilitate the automatic detection of fake or forged signatures on paper-based student attendance sheets. The current manual method requires teachers to compare signatures visually, which is a slow and subjective process that is likely to result in errors. Not only does this compromise the integrity of the academic institution, but it also increases the workload of the administration. FASDA is a solution that integrates image processing with deep learning–based offline signature verification, thereby guiding teachers in confirming attendance records.

The system is implemented as a web-based application that allows authenticated instructors to upload scanned attendance sheets, runs preprocessing and segmentation using OpenCV, and then performs verification using a Siamese CNN model trained on public and local datasets. A multi-layered structure divides the web interface, backend services, machine learning pipeline, and relational database. External connections (SheerID and Bank API) are utilized for verifying instructor identity and managing subscription. The design has laid out its main attractions as modularity (divergent ML service, preprocessing service, and web backend), scalability (model retraining and updating), and security (role-based access and audit logging) respectively.

2 Glossary

Term	***Definition***	***Source***
FASDA	*Fake Attendance Signature Detection Application; project aimed to understand how to detect fake signatures of students on attendance sheets via ML and image processing.*	*Project Definition*
Siamese CNN	*Architecture of neural networks consisting of two identical subnetworks which processes two images and produces a score which indicates how similar they are.*	*Hafemann et al. (2017)*
Signature Embedding	*Signature images are converted to texts which can be understood and manipulated mathematically to be able to do comparison of signatures.*	*Project Definition*
OpenCV	*Computer vision library that is freely available and can do a lot of things like binarization, contour finding, and image segmentation.*	*OpenCV Documentation*
TensorFlow	*One of the frameworks for deep learning which can do building and training and deployment of a model like the Siamese CNN which we did for verifying signatures.*	*TensorFlow Documentation*
Attendance Sheet	*A paper or digital document which is scanned or kept for recording attendance and has fields where students can write their names.*	*Project Definition*
Signature Container	*Region of the image that has been cropped to only show one signature and may or may not have information like student ID.*	*Project Definition*
Verification Result	*What the model thinks of a signature (whether it's real, fake, or some anomalous version) and how sure the model is about that.*	*Project Definition*
Final Report	*A report that has been automatically created that shows the signatures you uploaded in the attendance sheet and what the verification status was for each of them.*	*Project Definition*
BAYEK	*METU NCC's ethics committee responsible for approving biometric data collection procedures.*	*METU NCC Guidelines*

3 Architectural Views

In this report, the architecture of software systems will be described based on the Logical View, Process View, Development View, and Deployment View, as suggested by Kruchten (1995)[1]

3.1 Logical View

The logical view shows the key abstractions in the system as object classes.

3.1.1 Class Diagram

The FASDA class diagram portrays a modular and object-oriented system design divided into four major subsystems, namely: user management, course/attendance processing, ML-based signature verification, and payment/authentication services. Private attributes along with public operations are the means by which the encapsulation is enforced, and the class boundaries are kept clean. Inheritance (User -> Instructor/Admin) is used to express role specialization, while the relationship through composition (AttendanceSheet -> SignatureContainer) is in accordance with the actual data ownership. The classes, MLModel, ModelTraining and VerificationResult, are responsible for the signature-verification pipeline, while the external dependencies are handled through the implementation of lightweight client classes. This architectural design is a strong support for the system's maintenance, scalability, and unending alignment with the essential functional requirements of FASDA.

[1] P. B. Kruchten, "The 4+1 View Model of architecture," in IEEE Software, vol. 12, no. 6, pp. 42-50, Nov. 1995, doi: 10.1109/52.469759.

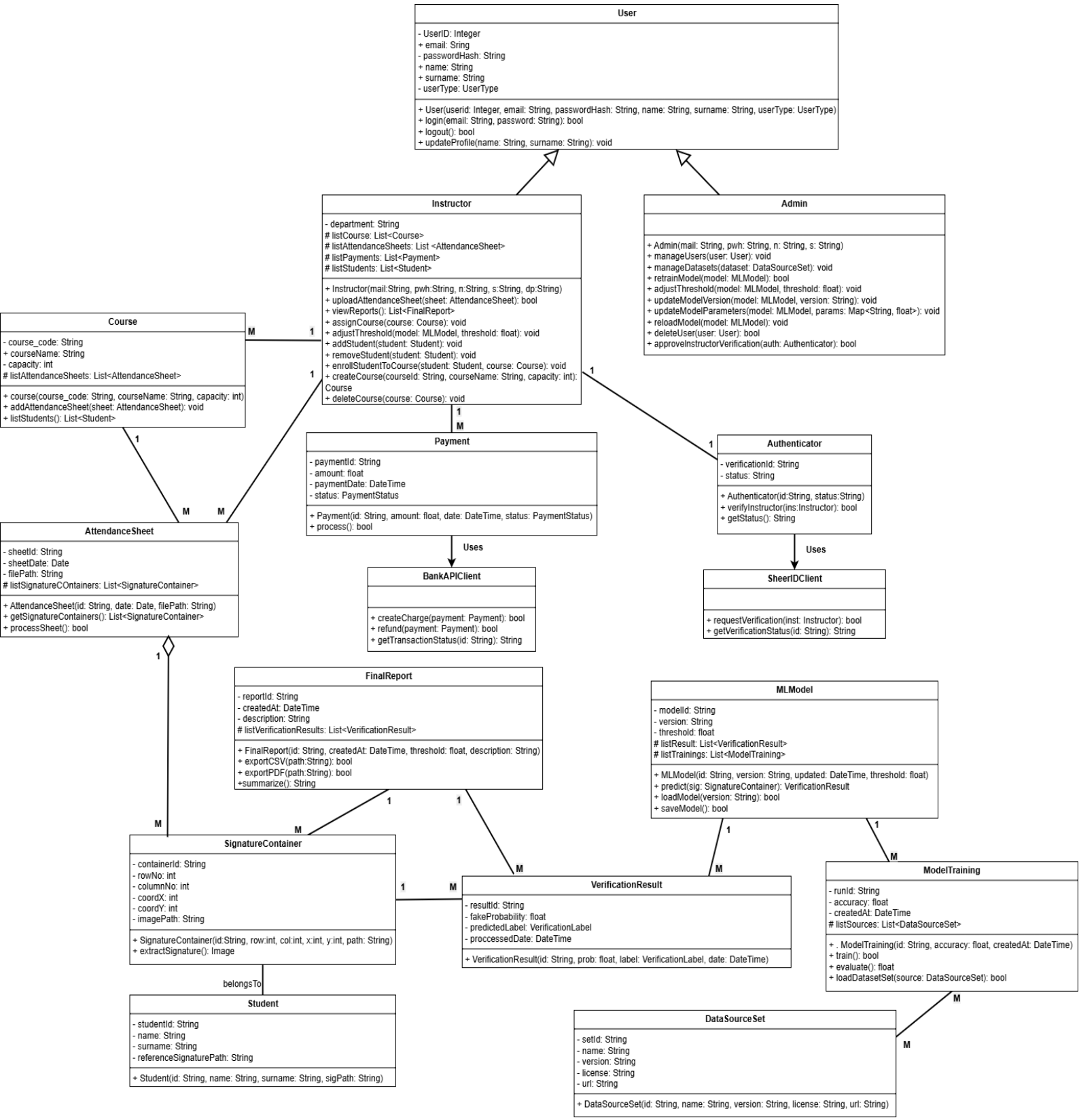

Figure 1. Class Diagram

Entities:

The users of the system, courses, attendance sheets, signature containers, ML verification components, payments, and authentication modules have all been incorporated into the system. Each entity is characterized by its own state and behavior.

Relationships:

Teachers are responsible for the courses and attendance sheets; attendance sheets contain signature containers; ML models generate verification outputs; educators are linked with payments and authentication records.

Purpose and Workflow Representation:

The diagram shows the entire signature verification process: data upload -> signature extraction -> ML prediction -> report generation. Each class represents one of the stages of this process and therefore the design is aligned with the actual system behavior.

External System Integration:

BankAPIClient and SheerIDClient are used to hide third-party services, thus the system can communicate with external payment and identity verification APIs without attaching the internal logic to the details of the implementation.

3.2 Process View

The process view shows how the system is composed of interacting processes.

3.2.1 Activity Diagram

- **Registration *Design Rationale:***

This activity diagram represents the workflow of onboarding an instructor and the client, backend system, authenticator, and payment system module. The flow contains vital steps including if an identity is verified and if the payment goes through which determines if an account is activated. This flow was selected as a critical diagram due to the risk of system verification, payment systems, system access, and the reliability of the onboarding process.

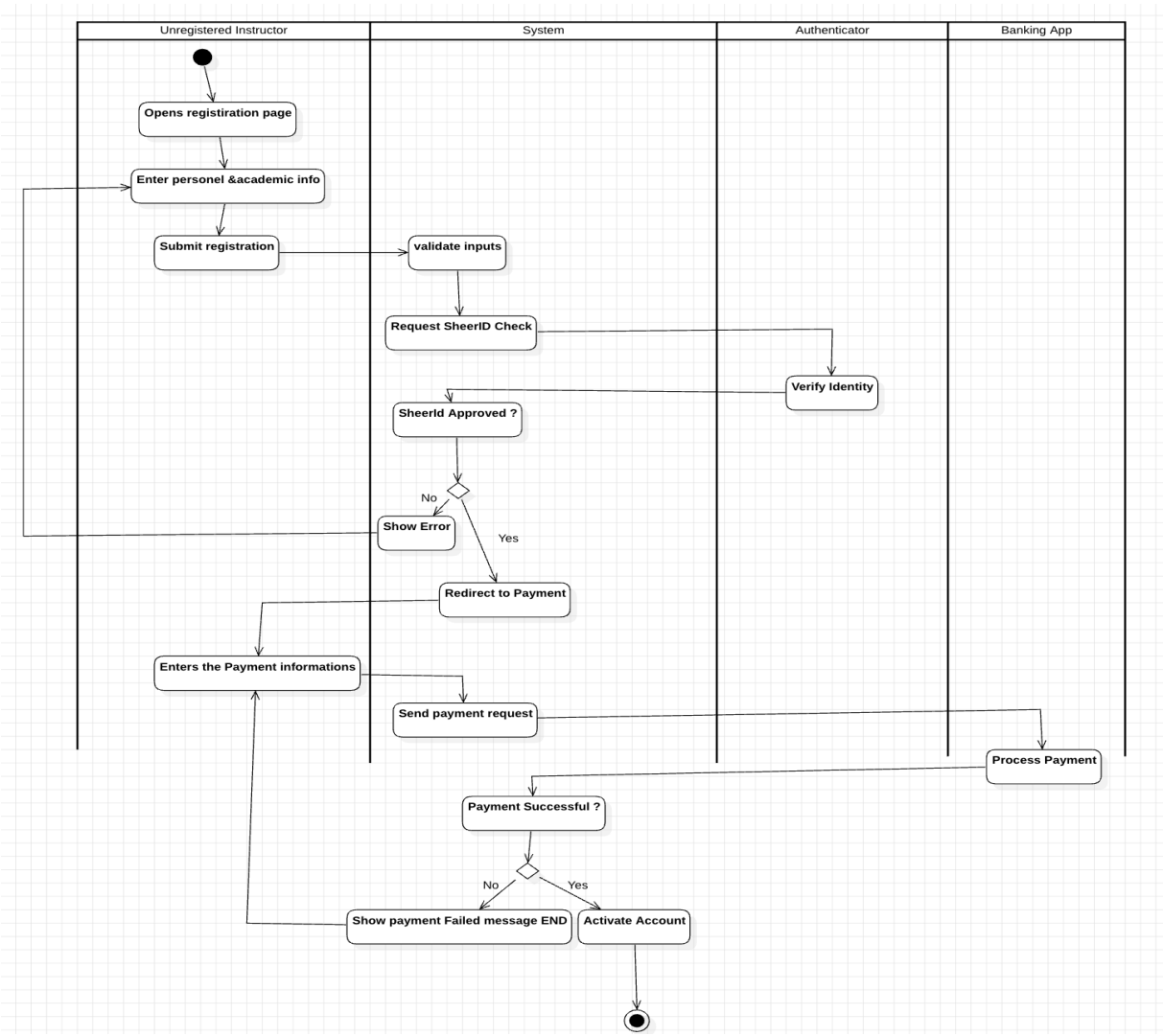

Figure 2. Registration Activity Diagram

- **Attendance Sheet Processing *Design Rationale:***

This diagram best exemplifies the automation of a prior tedious task: analyzing attendance sheets by running a batch of attendance sheets through a series of preprocessing and machine learning routines. The design partitions the roles of a Teacher, the Backend, the Preprocessor, the ML Classifier, and the Database. Each partition indicates the advancement of the data through the system with minimal human communication. The revision loop to illustrate the signature by signature processing and the control system was done to keep the diagram clean, while trying to illustrate the iterative machine learning nature of this signature processing. This flow is the most exemplary of the system capable of its primary value proposition: the extraction of the true and the forged signatures.

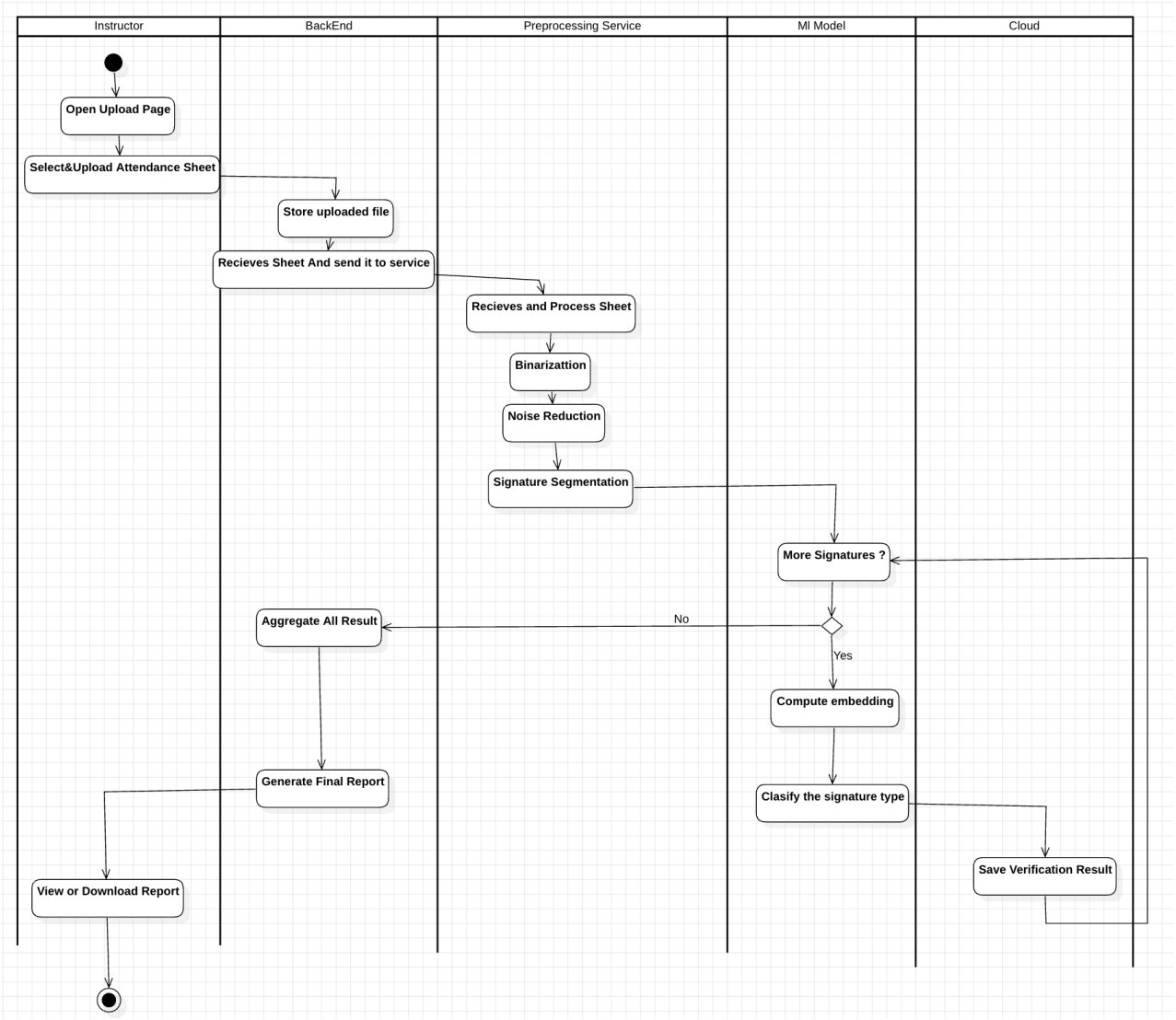

Figure 3. Sheet Processing Activity Diagram

- ***Flag Student Status Design Rationale:***

This diagram shows how users can manually verify and provide feedback where users examine results and manually flag questionable signatures. The diagram is kept simple on purpose and should concentrate on the decision point. This is where an end user examines the system's output and then causes an update to occur in the backend and database. Including this activity shows the human validation in the loop, an important component of the explanation of the automated ML pipeline. It shows how users modify the state of the system in an auditable manner, and ensures that the reported cases will continue to be available in subsequent iterations of the reports.

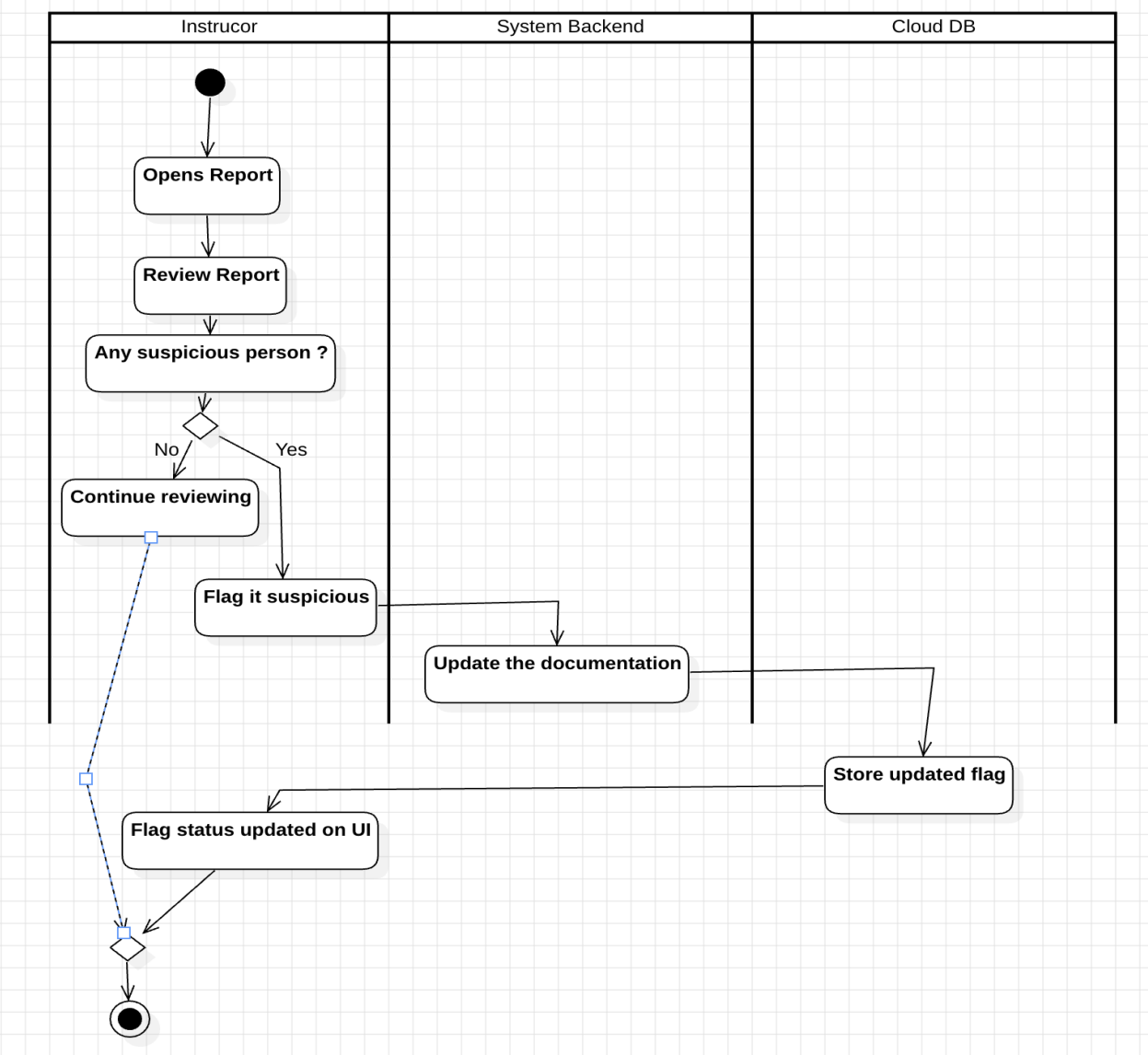

Figure 4. Flag Student Status Activity Diagram

3.2.2 Sequence Diagrams

The critical use cases of the FASDA system are illustrated through the sequence diagrams in this part. The main workflows have been presented here, following the guidelines of the SDD document, despite the fact that the system consists of numerous supporting and internal operations. For each of the critical use cases, we offer: a UML sequence diagram, an interaction table that corresponds to it, and a design rationale that justifies the architectural decisions made. The diagrams reveal the dynamic behavior across various system components without having the limitations of OOP structure.

SEQUENCE DIAGRAM 1 – Upload Sheet & Run Verification:

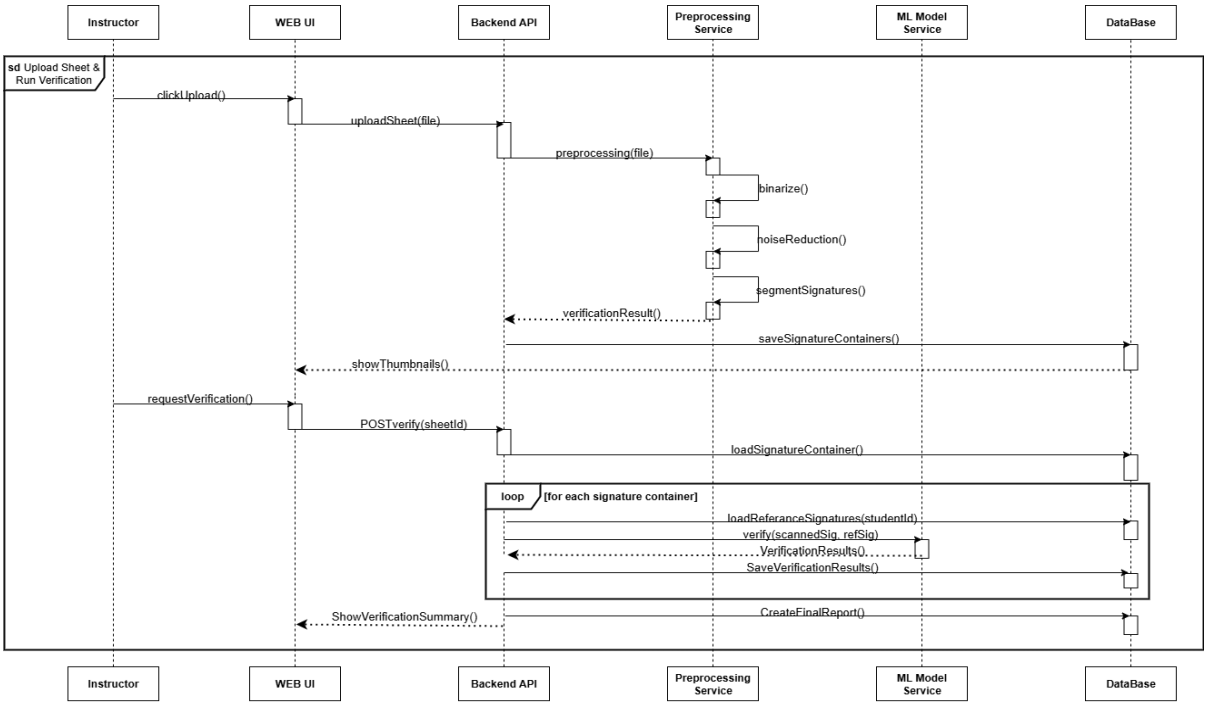

Figure 5. Upload Sheet & Run Verification Sequence Diagram

Use Case	Upload Sheet & Run Verification		
Actor	Instructor		
Use Case Controller	Controller		
Domain	DataBases	Screen	Other
FASDA		WEB UI	Preprocessing Service, ML Model Service
Step	Message	Owner	Parameters
1	clickUpload()	Instructor	-
2	uploadSheet(file)	WEB UI	file
3	preprocessing(file)	Backend API	file
4	binarize()	Preprocessing Service	-
5	noiseReduction()	Preprocessing Service	-
6	segmentSignatures()	Preprocessing Service	-
7	verificationResult()	Preprocessing Service	approved/denied
8	saveSignatureContainers()	Backend API	-
9	showThumbnails()	DataBase → WebUI	-
10	requestVerification()	Instructor	-
11	POSTVerify(sheetId)	WEB UI	sheetId
12	loadSignatureContainer()	Backend API	-
13	loadSignatureContainer() return	DataBase	containers
15	loadReferenceSignature(studentId)	Backend API	studentId
16	verify(scannedSig, refSig)	ML Model Service	signatures
17	verificationResult()	ML Model Service	result
18	SaveVerificationResult()	Backend API	result
19	saveVerificationResult return	DataBase	-
20	CreateFinalReport()	Backend API	sheetId
21	CreateFinalReport return	DataBase	report
22	ShowVerificationSummary()	Backend API → WEB UI	summary

Figure 6 . Upload Sheet & Run Verification Sequence Diagram Functionality

- ***Upload Sheet & Run Verification Design Rationale:***

  The FASDA system operates through its fundamental workflow which requires instructors to submit attendance records before starting the signature verification process. The solution operates through a modular design because user actions and result processing occur separately between the Web UI and Backend API, Preprocessing Service, ML Model Service, and Database provider. The loop structure allowed us to handle signature container separately, consequently, it will make it easier to scale and troubleshoot. This makes it possible to scale and debug the system more easily. The verification pipeline processes large sheets quickly through this architecture which maintains component uniformity throughout the system.

SEQUENCE DIAGRAM 2 – Retraining ML Model

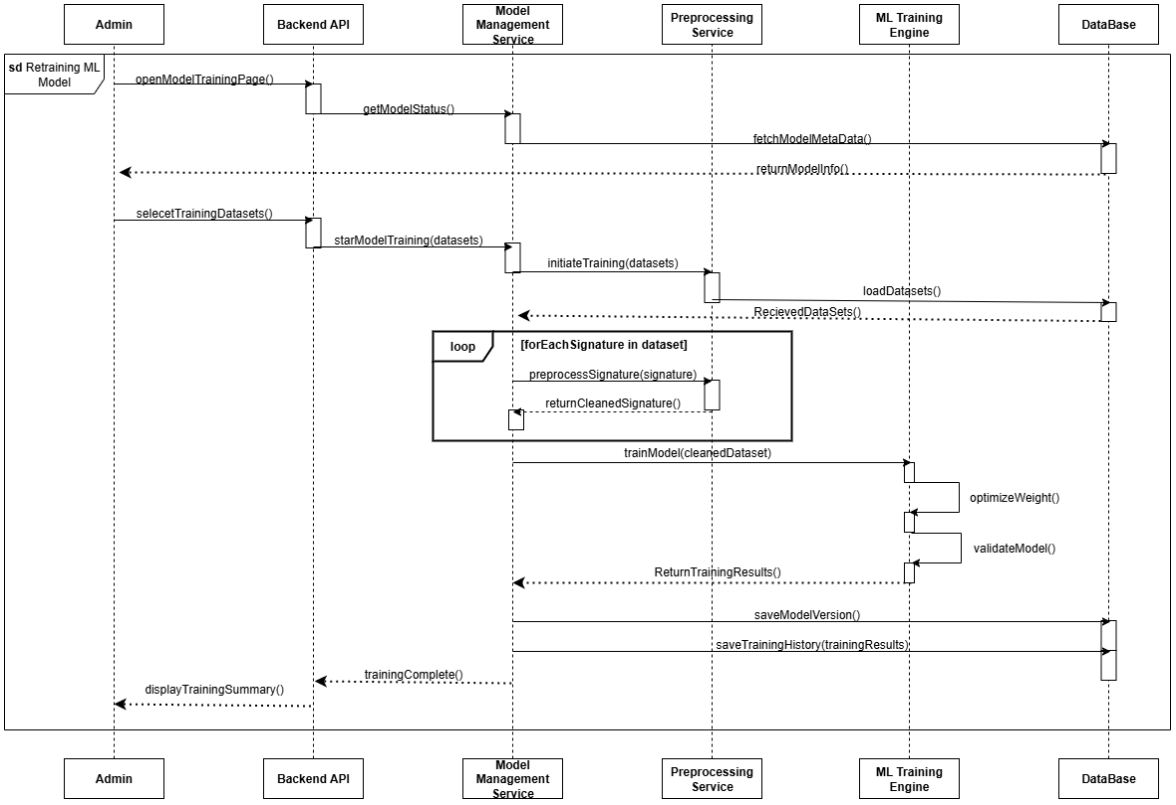

Figure 7.Retraining ML Model Sequence Diagram

Use Case	Retraining ML Model		
Actor	Admin		
Use Case Controller	Controller		
Domain	DataBases	Screen	Other
FASDA	DataBase	AdminModelScreen	Model Management Service, Preprocessing Service, ML Training Engine
Step	Message	Owner	Parameters
1	openModelTrainingPage()	Admin	-
2	getModelStatus()	Backend API	-
3	fetchModelMetaData()	Model Management Service	-
4	returnModelInfo()	DataBase	modelMetadata
5	selecetTrainingDatasets()	Admin	-
6	starModelTraining(datasets)	Backend API	datasets
7	initateTraining(datasets)	Model Management Service	datasets
8	loadDatasets()	Preprocessing Service	-
9	RecievedDataSets()	DataBase	dataset
11	preprocessSignature(signature)	Model Management Service	signature
12	returnCleanedSignature()	Preprocessing Service	cleanedSignature
13	trainModel(cleanedDataset)	Model Management Service	cleanedDataset
14	optimizeWeight()	ML Training Engine	-
15	validateModel()	ML Training Engine	-
16	ReturnTrainingResults()	ML Training Engine	result
17	saveModelVersion()	Model Management Service	-
18	saveTrainingHistory(trainingResults)	Model Management Service	trainingResults
19	trainingComplete()	Model Management Service	-
20	displayTrainingSummary()	Backend API	summary

Figure 8.Retraining ML Model Sequence Diagram Functionality

- ***Design Rationale:***

  This sequence diagram illustrates the process by which FASDA goes about retraining the ML model when signatures data states are collected. The system has separated the steps of the process such as loading the dataset, fitting on samples, training the model, and testing the outputs in sections, to allow for better organization and updates in the future. The loop highlights how each new signature sample is probed for one single signature sample before model entraining, which helps the model keep its input clean and consistent whenever new samples come into the process. In the end, the needed updates to the model and the training history will be saved. This allows us to have a record of the changes that can track improvements for future models.

SEQUENCE DIAGRAM 3 – User Registration

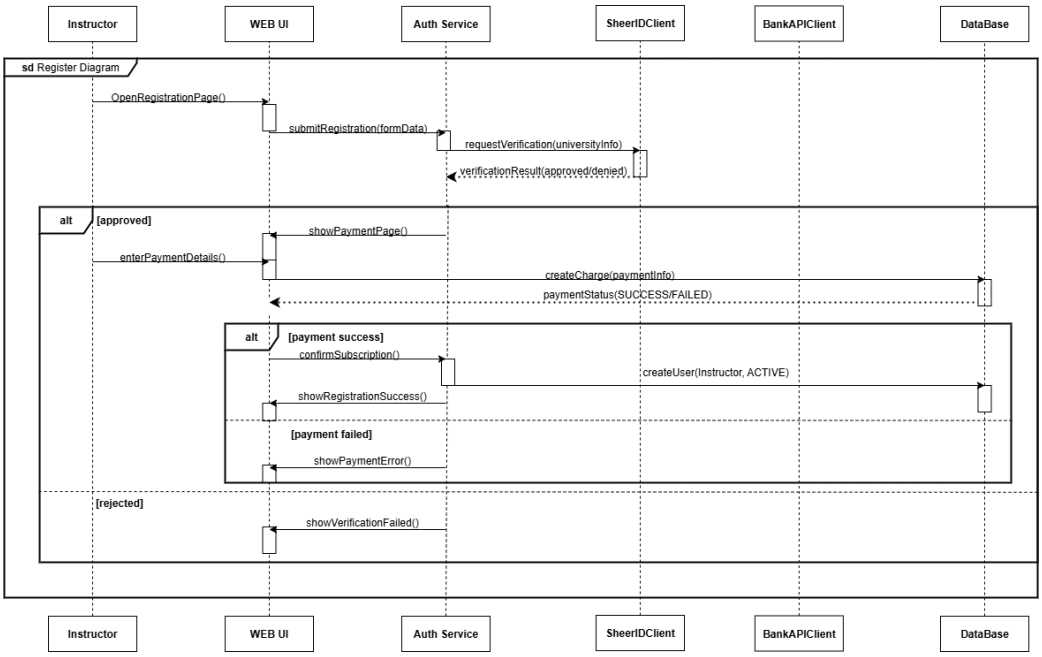

Figure 9. User Registration Sequence Diagram

Use Case	User Registeration		
Actor	Instructor		
Use Case Controller	Controller		
Domain	**DataBases**	**Screen**	**Other**
FASDA	User DB	RegisterScreen	SheerIDClient, BankAPIClient
Step	**Message**	**Owner**	**Parameters**
1	OpenRegistrationPage()	Instructor	-
2	submitRegistration(formData)	WEB UI	formData
3	requestVerification(universityInfo)	Auth Service	universityInfo
4	verificationResult(approved/denied)	SheerIDClient	status
5	enterPaymentDetails()	WEB UI	paymentInfo
6	showPaymentPage()	Auth Service	-
7	createCharge(paymentInfo)	BankAPIClient	paymentInfo
8	paymentStatus(SUCCESS/FAILED)	BankAPIClient	status
9	createUser(Instructor, ACTIVE)	Auth Service	userData
10	saveUserRecord	DataBase	userData
11	showRegistrationSuccess()	Auth Service	-
12	showPaymentError()	Auth Service	-
13	showVerificationFailed()	Auth Service	-

Figure 10. User Registration Sequence Diagram Functionality

- ***Design Rationale:***

  This sequence diagram illustrates the onboarding process instructors will go through before being able to engage with FASDA. In the onboarding process, instructors will undergo identity verification (which will utilize SheerID) and payment processing (using the Bank API). This was done to make sure that only authenticated and subscribed instructors can access the system. The Auth Service will manage all backend processing while user interaction will be managed by the Web UI when moving through the onboarding steps. This structure in the onboarding process separates concerns, enhances the overall security of the system, and verifies to the system that the instructor is who they say they are prior to continuing the onboarding process.

3.2.3 Data Flow Diagrams

Level o:

The Level o design separates the responsibility for managing Admin Models from the responsibilities for Instructors . The two stores used for storing and accessing the data are a common user/flag data store (D1) that stores all of the information about users and flags in one central location, and a dedicated Trained Model Data Store (D2), which is set up to maintain the integrity of the model and allow for separate updates to the model.

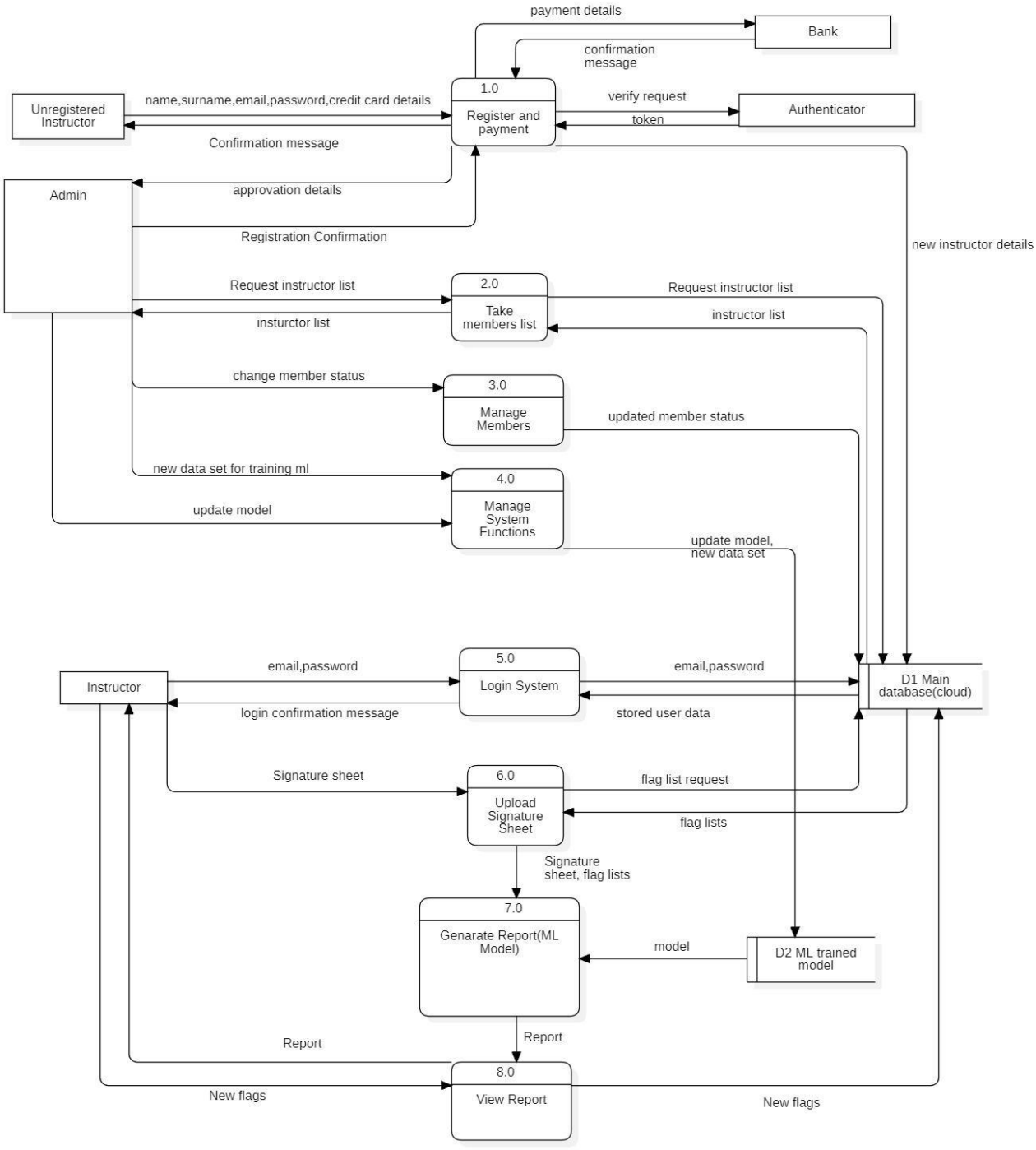

Figure 11. level-o Data Flow Diagram

3.3 Development View and Physical View

The development view shows how the software is decomposed for development, and the physical view shows the mapping of software onto hardware.

3.3.1 Component Diagram and Deployment Diagram

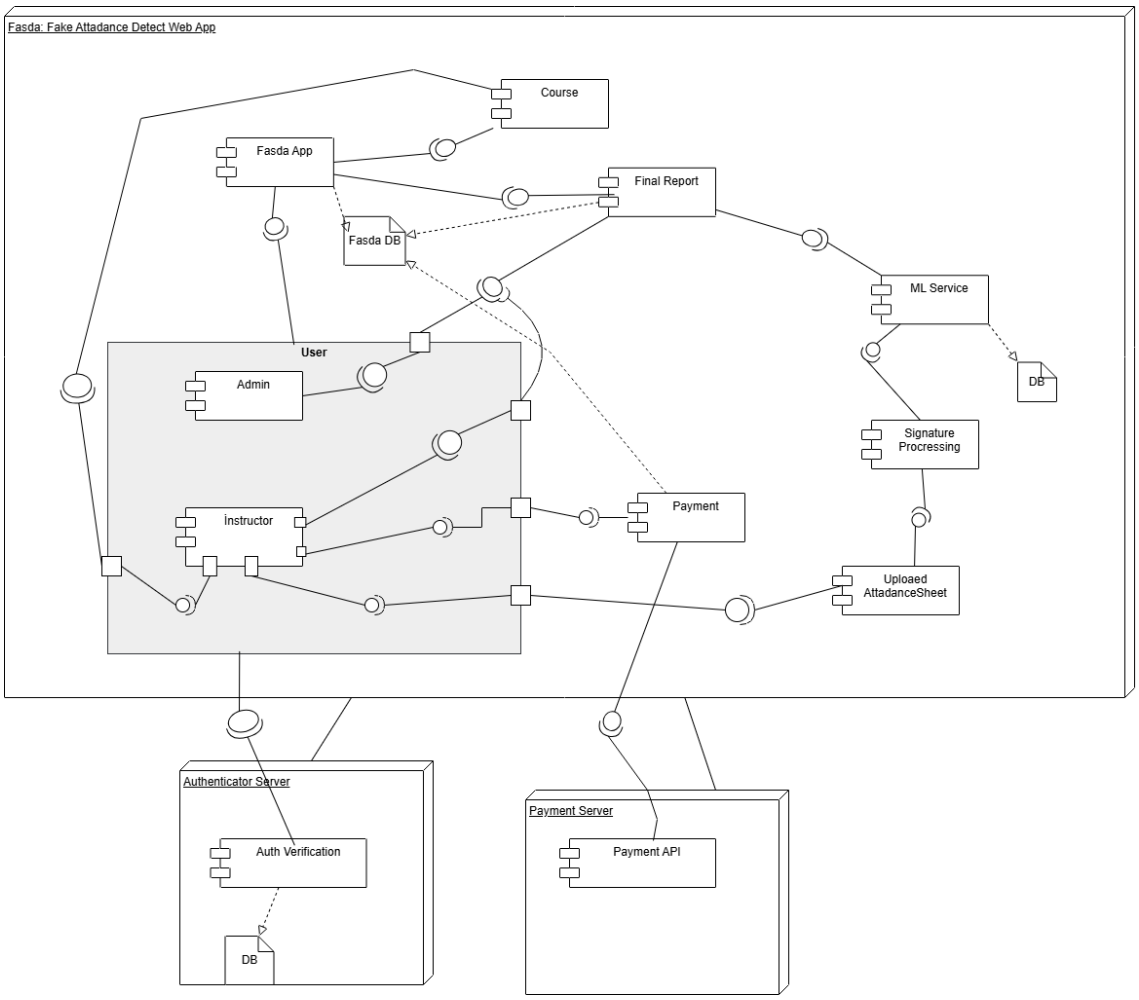

Figure 12. Component & Deployment Diagram

- The diagram combines UML Component and Deployment views in a one figure, internal components are (Fasda App, Course, UploadAttendanceSheet, SignatureProcessing, ML Service, Final Report, Payment, User "Admin and İnstructor") are components on the FASDA Web Server, while external services are (Authenticator Server and Payment Server) are shown as external nodes.
- Course and UploadAttendanceSheet are relative to course and attendance–sheet classes.
- User Components divide into Admin and İnstructor. Users can view final report on system.
- Final Report is the general report for attendance sheet which is uploaded by İnstructor on system.
- The relationship from ML Service to Final Report is with verificaton result component, but on the diagram we assume that ML service is includes that component. Verification result is creating by ML service, which is analysis the attedance sheet data and detect the forgein signature or not.

4 References

FASDA Project Team. (2025). Fake Attendance Signature Detection Application – Project Definition Document. Middle East Technical University, Northern Cyprus Campus.

Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2017). Learning features for offline handwritten signature verification using deep convolutional neural networks. Pattern Recognition, 70, 163–176. https://doi.org/10.1016/j.patcog.2017.05.012

FASDA Project Team. (2025). Signature Embedding Specification. Internal Documentation, METU NCC.

OpenCV. (2024). OpenCV Documentation. https://docs.opencv.org/

TensorFlow. (2024). TensorFlow Documentation. https://www.tensorflow.org/

Middle East Technical University Northern Cyprus Campus. (2024). BAYEK Ethical Guidelines for Biometric Data Collection and Processing. METU NCC Research Ethics Committee.

5 Appendices

5.1 Project Scheduling

5.1.1 Milestones and Tasks

ID	Milestone / Task	Estimated Duration (Weeks)	Depends On	Responsible
M1	Project Definition & Requirement Finalization	1	-	All team
T1	Complete SRS Document	2	M1	Bora, Alperen
T2	Prepare Class Diagrams & Architecture Draft	1	T1	Ali Efe, Burak
M2	SDD Architecture Freeze	1	T2	All team
T3	Activity, Sequence & DFD Diagrams	1	M2	Bora, Ali Efe
T4	Component & Deployment Diagrams	1	M2	Burak
T5	Dataset Search & Licensing	2	-	Bora
T6	ML Pipeline Research (Siamese, Segmentation)	3	T5	Bora, Alperen
M3	ML Design Completion	1	T6	All team
T7	Preprocessing Module Draft Implementation	2	M3	Alperen
T8	Initial Model Training Attempt	2	T7	Bora
T9	Web Backend Routing & API Structure	2	M2	Ali Efe
M4	Sprint 2 Completion	1	All prior tasks	All team
T10	Integration of ML & Backend	2	M3, T9	Bora, Ali Efe
T11	Report Generation Module	1	T9	Burak
M5	Final System Review	1	T10, T11	All team

5.1.2 Gantt Chart

Provide a Gantt Chart that shows both the milestones and tasks specified in Section 5.1.1

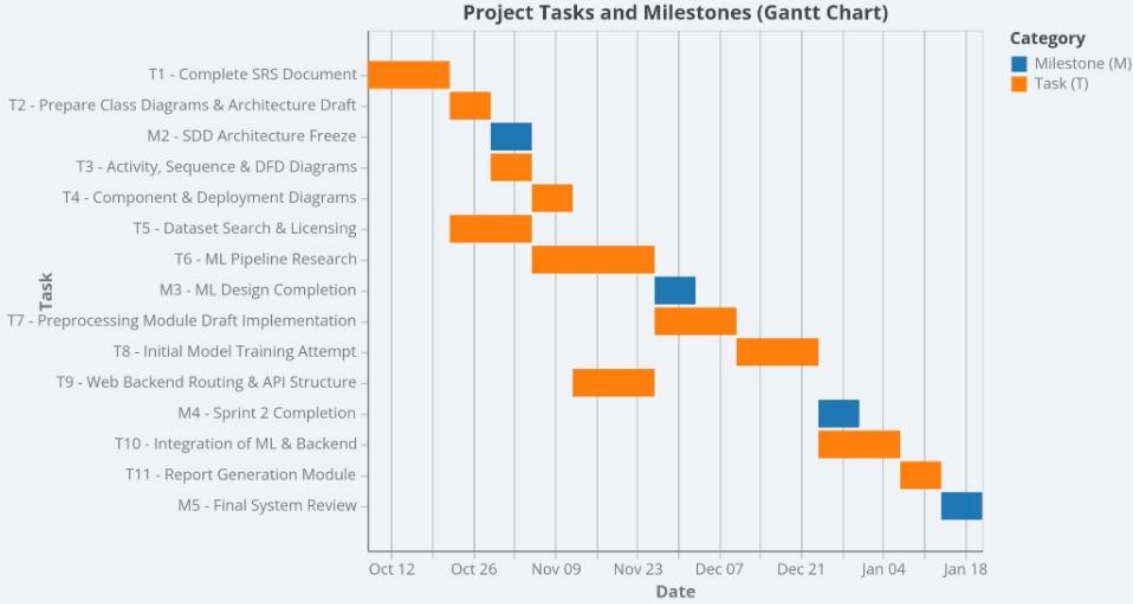

Figure 13. Gantt Chart

5.2 Sprint 2 Overview

5.2.1 Sprint Backlog

List the tasks completed in the sprint and the time spent for each in hours. Do not need to write them in the form of user stories.

Task	Description	Time Spent (Hours)	Responsible
SB1	Activity Diagrams (3 diagrams)	6 hrs	Bora
SB2	Sequence Diagrams (3 diagrams)	5 hrs	Ali Efe
SB3	Data Flow Diagrams	4 hrs	Burak
SB4	Component & Deployment Diagrams	4 hrs	Burak
SB5	Class Diagram Revision & Updates	3 hrs	Alperen
SB6	Architecture Review Meeting	2 hrs	All Team
SB7	SDD Document Editing & Formatting	5 hrs	Bora

5.2.2 Sprint Burndown Chart

The chart should show the remaining time in hours of the tasks of the sprint over the weeks.

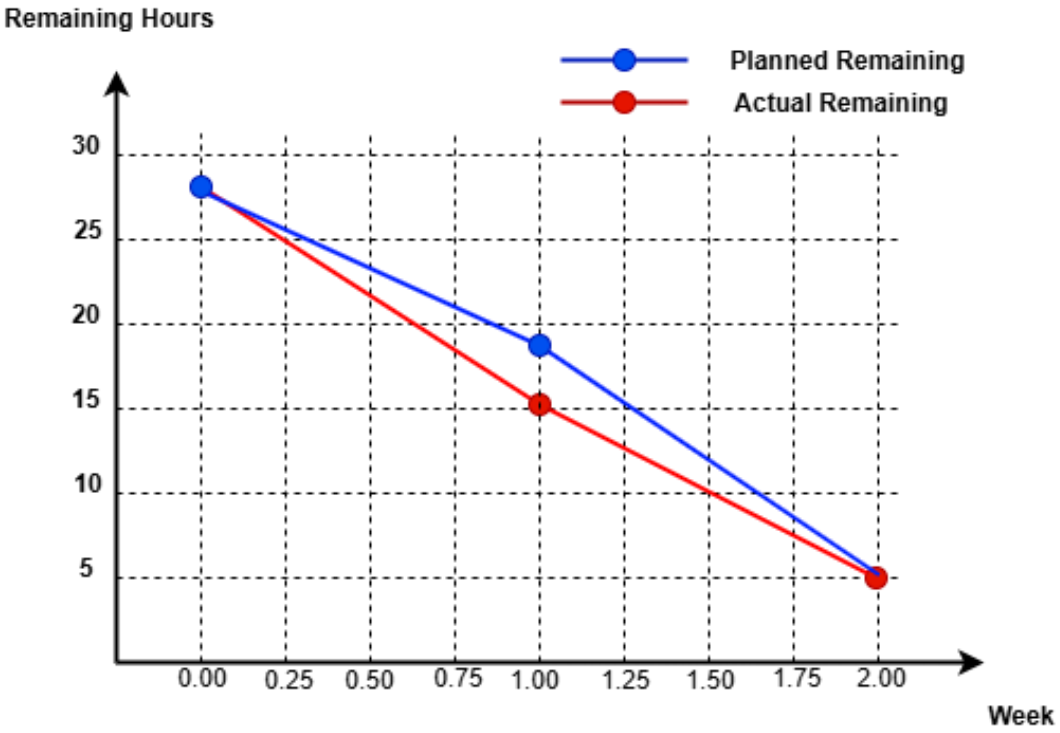

Figure 14. Sprint Burndown Chart

5.2.3 Sprint Review

During this sprint, the necessary database configurations were implemented, and data extraction from GPDS was successfully completed. Concurrently, the petition for the Ethics Committee application was drafted to address administrative requirements. While research into Siamese Networks for the ML module continues, the lack of response from certain external database providers has caused a minor bottleneck in dataset expansion; however, aside from this external delay, no significant issues were encountered throughout the sprint. Additionally, SDD has been successfully finalised.

5.2.4 Sprint Retrospective

This project's Sprint included evenly dividing the tasks for the Software Design Document (SDD) among group members. While there were some minor communication issues within the group due to students' focus on midterm exams at the time of this Sprint, coordination was achieved through good communication between all participants. In addition to the documentation, the assigned group members also accomplished all technical tasks related to the Database, Ethics Committee Application, and ML Modeling. Thus, the critical Infrastructure for the project is now almost complete in anticipation of the next phase of the project: Application Implementation.