2005

A Web-based environment for automated dental identification research

SatyaSrinivas Chekuri
West Virginia University

A WEB-BASED ENVIRONMENT FOR AUTOMATED DENTAL IDENTIFICATION RESEARCH

by
SatyaSrinivas Chekuri

Thesis
Submitted to
The College of Engineering and Mineral Resources
at
West Virginia University
In partial fulfillment of the requirements for the degree of
Master of Science
in
Electrical Engineering

Dr. Hany Ammar, Ph.D., Chair
Dr. Xin Li, Ph.D.
Dr. Arun Ross, Ph.D.

Lane Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2005.

Keywords: Dental Identification, ADIS, 3-Tier Architecture, Law Enforcement, Digital Government.

ABSTRACT

A Web-Based Environment for Automated Dental Identification Research

SatyaSrinivas Chekuri

The Criminal Justice Information Services (CJIS), division of the Federal Bureau of Investigations (FBI), include in its strategic plan the creation of an Automated Dental Identification System (ADIS), a Post Mortem Dental Identification System. In past few years researchers from West Virginia University, Michigan State University and University of Miami are working to improve the accuracy and performance on different realizations developed fro ADIS. Apart from these features there was also a need to make this system accessible through web. Web-ADIS is a web-interface for Automated Dental Identification Research.

This project aims at designing an end-to-end web-interface to meet the requirements of ADIS like Identification, Maintenance and Bridge Modules. In Identification Mode the subject record will be uploaded by the user and the match list is obtained as result. Maintenance Mode enables uploading of reference records and to populate the database with preprocessing data. Bridge Module enables researchers from other universities to use the database designed in WVU. A database is also designed to hold non-dental features like name, age, gender etc and dental features like preprocessing data.

This provides the FBI agents and the Forensic experts the ability to use ADIS from their office desks. This web interface provides an Identification Module, Maintenance Module and Bridge Module. Future work involves developing a testing module which will help researchers to test the different realizations to examine their results and System Configuration to configure the system to use realizations of user's choice for identification.

ACKNOWLEDGEMENTS

I wish to thank my supervisor Dr. Hany Ammar for his constant support and guidance for my work and on guidance prior to this thesis.

I would like to extend my thanks to Dr. Arun Ross and Dr. Xin Li for serving on my committee and also for their encouragement and support.

I would also like to thank Diaa and members of ADIS group for their continuous, unconditional help and support during this project.

I would like to thank my parents, sister, brother-in-law and all other relatives for standing by me and blessing me for success without which I would not have reached so far. I would also like to thank my friends who were with me and extended their helping hand in times of need.

Table of Contents

List of Tables

List of Figures and Illustrations

CHAPTER 1

INTRODUCTION

In this is chapter we introduce the basic definitions of biometrics, forensic identification systems followed by brief description of Automated Dental Identification Systems (ADIS) and web interface for Automated Dental Identification System (web-ADIS) which is again followed by motivation and scope of research. In sections 1.1 and 1.2 we give an overview of biometric identification systems. Section 1.3 gives an overview about Automated Dental Identification System followed by brief description of Web-ADIS in section 1.4. The motivation for research is discussed in section 1.5. Section 1.6 describes the way this thesis is organized.

1.1 Biometrics:

Authentication methods such as username/password, PIN were mostly used in early 20th century. Increased security concerns in the present world saw development of wide variety of authentication methods which assures more accuracy and convenience. These methods are dominated by biometric technologies. Biometrics refers to the automatic identification of individuals based on their physiological and behavioral characteristics [Jain99]. The different verification techniques include face, finger prints, hand geometry, dental features, hand writing, iris, retinal, vein and voice.

1.2 FORENSIC IDENTIFICATION SYSTEMS:

Forensic identification is another branch of science that uses natural sciences to help solve several legal cases and public issues [RM03]. This is mainly used to study the crime scene to zero on the criminal or to identify the victim. The biometric features like fingerprints, DNA, hair, blood, tissues and dental features are used in forensic identification.

Forensic identification is used mainly for

- Anti Mortem (AM) Identification
- Post Mortem (PM) Identification

Anti Mortem identification is used to identify the people who are still alive. Fingerprints, DNA are commonly used biometric features for identification. These biometric features are taken from suspects and matched with that collected from crime scene to identify the main criminal.

Post Mortem identification is used to identify the person after his death. This was a challenge faced by forensic experts from decades. The identification of an individual is difficult due to the extent of injury or degree of decomposition of the body. The biometric features like finger prints are unreliable due to body transition the extent of which depends on the time elapsed since death and several other condition like temperature,

humidity, wind and many other factors [Joseph]. Dental features are reliable biometric features for post mortem identification in most cases.

The goal of the dental task force (DTF) created by the Criminal Justice Information Services Division (CJIS) is to improve the utilization and effectiveness of the National Crime Information Center's (NCIC) Missing and Unidentified Persons (MUP) files. CJIS recommended the creation of an Automated Dental Identification System (ADIS) with goals and objectives similar to the Automated Fingerprint Identification System (AFIS) but using dental characteristics instead of fingerprints [Zanab04].

1.3 ADIS:

ADIS is an Automated Dental Identification System which takes a subject Post Mortem (PM) dental record as an input and compares it with the Anti Mortem (AM) dental records in the Digital Image Repository (DIR) and gives a list of AM dental records which are similar to subject record.

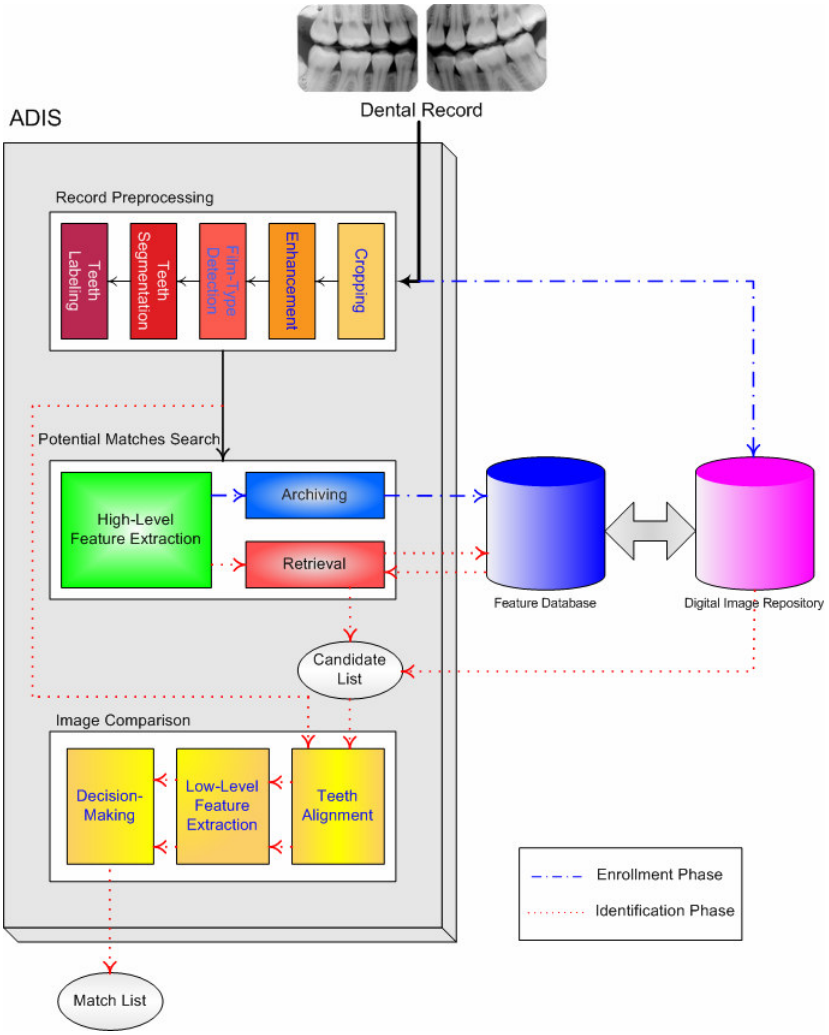


Figure 1.1: Block diagram of Automated Dental Identification System.

ADIS is an assortment of the following mega-components (as depicted in Figure 1.1):

(i) Record Preprocessing component

(ii) Potential Matches Search component

(iii) Image Comparison component.

The record preprocessing component handles cropping the records into dental films, enhancement of films to compensate for possible poor contrast, classification of films into bitewing, periapical or panoramic views, segmentation of teeth from films and annotating teeth with labels corresponding to their location.

The potential match search component manages archiving and retrieval of dental records based on high-level dental features (e.g. number of teeth and their shape properties) and produces a candidate list.

The image comparison component mounts for low-level tooth-to-tooth comparison between subject teeth after alignment and the corresponding teeth of each candidate, thus producing a short match list. The philosophy behind architecting the prototype ADIS as such is to exploit high-level features for fast retrieval of a candidate list produced by the potential matches search component and then to refine the candidate list using low-level image features thus producing a short match list [IBIMA]. This match list is sent finally to a forensic expert to get find the correct match.

DIR which can be seen on the right in the block diagram is a remote Database Server where image files and radiographs are stored. Feature Database is a Database server which has textual Dental codes. The data in DIR and NCIC Repository is used in several

stages like potential match to get a list of probable matches called Candidate List and image comparison to get the final list called Match list.

1.4 Web-ADIS:

Web-ADIS is a web interface for Automated Dental Identification System. The operations of web-ADIS are mainly classified into three different modules.

- Configuration Module
- Identification Module
- Maintenance Module

Configuration Module:

The research on ADIS is being done in three universities West Virginia University, Michigan University and University of Miami. There are various realizations that are being developed on preprocessing components and potential components.

The configuration module helps the user to select a realization from the list of available realizations. The parameters and their respective parameter values can also be selected for the selected realization. The different components in this module are preprocessing component, Potential match component and Image Comparison component. Preprocessing component consists of sub-components like record cropping, film type classification, film enhancement, teeth segmentation, teeth labeling. Potential match

component consists of record retrieval as its sub-component. Image comparison component consists of alignment, normalization, compression, micro-decision, macro-decision as its sub-components.

The module allows the user to save the configuration for future use. A rookie user can use a default configuration, the configuration of which is selected before hand as a default by an expert.

Identification Module:

This module is used to identify the PM subject record. The subject record has to pass through three stages Preprocessing, Potential Match and Image Comparison. At the end of Image Comparison a list of AM records from the DIR which are in close resemblance to subject record is generated. These records are sent to forensic experts to get the exact match.

Maintenance Module:

This module helps the user to maintain the database. The maintenance module can be used to upload reference records into DIR, update or replace the image processing algorithms with better ones, update NCIC information for existing reference records.

1.5 MOTIVATION

We are motivated to develop a web interface for the automated dental identification system to help different forensic experts from all over the world access our services. This will save forensic experts the trouble to examine all the reference records before getting the match for the subject record. Through this interface Configuration, identification and Maintenance modules are designed and developed by using Object-Oriented Approach.

1.6 THESIS ORGANISATION

A brief summary of the topics discussed in chapters are give below:

Chapter 1: This chapter gives an introduction to Automated Dental Identification System and Web-ADIS. This introduces modules which form the skeleton of Web-ADIS.

Chapter 2: This chapter gives background information on two Computer Aided PM Identification Systems WinID and CAPMI. This also describes the following technologies: the three tier architecture of J2EE, Java, JSP, Java Beans, JDBC and SQL.

Chapter 3: This chapter presents the problem statement and the research objectives.

Chapter 4: This chapter describes the components of Web-ADIS. This describes the various important modules in Web-ADIS and their behaviour with block diagram.

Chapter 5: This chapter presents the design and implementation of Web-ADIS using class diagrams.

Chapter 6: This chapter describes the test scenarios, problems faced while testing the components of Web-ADIS and methods used to solve them. This chapter also describes the problems in Web-ADIS which remained unsolved.

Chapter 7: This chapter suggests future work that can be done in Web-ADIS.

CHAPTER 2

Background Work

2.1 Introduction:

The implementation of the proposed environment is based on J2EE and SQL. Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing component-based multitier enterprise applications. J2EE simplifies building enterprise applications that are portable, scalable, and that integrate easily with legacy applications and data [java].The different J2EE technologies used in this implementation are Java Server Pages (JSP), Java Beans, Java Script, Java Native Interface (JNI).

Section 2.1.1 explains about two computer-aided PM identification systems WinId and CAPMI. In section 2.1.2 three tier environment implemented in J2ee is discussed followed by JavaServer Pages, Java DataBase Connectivity, JavaBeans, Java Native Interface and JavaScript in sections 2.1.3, 2.1.4, 2.1.5, 2.1.6 and 2.1.7 respectively. SQL used for implementing the database design in Oracle is explained in section 2.2.

2.1.1 Computer Aided PM Identification Systems [Diaa]:

The most famous among these systems are CAPMI and WinID. However, these

identification systems do not provide a high level of automation, as feature extraction, coding, and image comparison are still carried-out manually.

Computer Assisted Post Mortem Identification system (CAPMI):

The Computer Assisted Post Mortem Identification system (CAPMI) was developed by the bioengineering branch of the US Army Institute of Dental Research. CAPMI is a computer software program that compares between dental data extracted from AM and PM dental records. The program generates a prioritized list of candidates based on the number of matching dental characteristics.

The list guides forensic odontologists to the most probable AM matches to the submitted PM records, so that they can rapidly proceed with a positive identification via comparison of radiographs. In CAPMI, data entry is performed either through keyboard input or an optical mark reader, the dental information associated with each tooth include:
Missing tooth, Unerupted tooth, Cavity on tooth, Anomalous condition, Mesial restoration, Occlusal restoration, Porcelain jacket crown, Facial restoration, Lingual restoration, Full coverage crown, Amalgam restoration, Stainless steel crown, Non-metallic restoration ceramic or acrylic/metal, Distal restoration, Gold/cast metal restoration, Temporary restoration, ¾ crown, Pointic root canal, Treated tooth, Removable partial denture, Deciduous tooth, Virgin tooth.

CAPMI uses up to 16-bits of information per tooth for dental data it represents. For instance, if a tooth is present bit '0' is set otherwise it is reset. The other bits are used to represent the 5 surfaces of the tooth and to indicate if a crown is present or the tooth is root filled. The following textual notation is used to describe the characteristics of each tooth: A mesial cavity is entered as (m), a crown is entered as (jmodbp) to indicate that all five surfaces are restored (modbp) and a crown is present (j). The presence of a tooth is indicated by (.) and its absence is indicated by (x). Thus UL6.mod indicates that the Upper Left first permanent molar is present and has restorations in the mesial, occlusal and distal surfaces, while LL3 x indicates the Lower Left canine is missing. (b: bucal, p: palatal).

After entering the postmortem data for all the teeth, dental information of the subject(s) is compared to the dental information of the candidate(s). The result is a list of the most likely matches, sorted in order of the most likely match to the least likely match. The comparison output is either: Match, Mismatch, or a Possible match. A Match occurs when the AM and PM dental data are exactly the same. A Mismatch occurs when AM dental condition is different from the PM dental condition given that it could have not evolved into the PM dental condition (e.g. a tooth with AM three restored surfaces can not evolve into a PM two restored surfaces tooth). A Possible match occurs when the AM dental condition is different from the PM dental condition but could have evolved into the PM condition (e.g. a tooth with AM two restored surfaces may evolve into a PM three restored surfaces tooth).

The matching criteria are chosen such that resulting candidate lists satisfy certain accuracy requirements in terms of false acceptance rate (FAR) and false rejection rate (FRR). Then the forensic experts examine the dental radiographs of the candidates for identification purposes. CAPMI is not considered an identification system, it merely provides a list of most likely identities for use by the forensic team. This list precludes the need for an extensive manual comparison of many antemortem to the postmortem records.

According to a study carried-out by the US Army on 7030 Army males (mean age of 24 years), the average individual has seven dental characteristics. For individuals with four or more characteristics, 93% had no other person with the same characteristics. For individuals with 7 dental characteristics, the correct match was the top of the candidate list in 95% of the trials and a correct match was found within the first ten records 100% 17 of the time. CAPMI is capable of comparing and sorting 1200-5000 records per second on most personal computers.

WinID:

WinID is a dental computer system that matches missing persons to the unidentified persons using dental and anthropometric characteristics to rank possible matches. Other information about physical descriptors, and pathological and anthropologic findings can be fed to WinID database. The dental codes used in WinID are extensions to the CAPMI codes. Up to five primary codes may be specified and a dash (-) is placed between the

primary and the secondary codes, however, most of the searches in WinID utilize only the primary codes. Following is a listing of the primary and secondary codes used in WinID.

WinID Primary Codes:

- **M: mesial** surface of tooth is restored.
- **O: occlusal** surface of posterior tooth is restored.
- **D: distal** surface of tooth is restored.
- **F: facial** surface of tooth is restored.
- **L: lingual** surface of tooth is restored.
- **I: incisal** edge of anterior tooth is restored.
- **C:** tooth is fitted with a **crown**.
- **U:** tooth is **unerupted**.
- **V:** non-restored tooth – **virgin**.
- **X:** tooth is missing – **extracted**.
- **J:** tooth is **missing postmortem** or the clinical crown of the tooth is not present for examination. Also used for avulsed tooth. The root or an open socket is present, but no other information is available.
- **/: no information** about tooth is available.

WinID Secondary Codes:

- **B:** tooth is **deciduous**.
- **E: resin** filling material.
- **G: gold** restoration.
- **H: porcelain**.
- **R: root** canal filled.
- **S: silver** amalgam.
- **A: anomaly** is associated with this tooth. Specifics of the anomaly may be detailed in the comments section.
- **N: non-precious** filling or crown material. Includes stainless steel.

• **P: pontic**. Primary code must be X to indicate missing tooth.

• **Q: three quarter** crown. Primary code must be C to indicate crown.

• **T: denture tooth**. Primary code must be X to indicate missing tooth.

• **Z: temporary** filling material. Also indicates gross **caries**.

Figure (2.8) shows an example of encoding and charting dental data using the WinID software for the bite-wing film pair on the upper right corner of the figure. Each tooth is assigned a two-digit number, the most significant digit indicates the jaw segment in which the tooth lies, while the second digit indicates the position of the tooth in its jaw segment. The right side of the upper jaw is assigned '1', the left side of the upper jaw is assigned '2', the left side of the lower jaw is assigned '3', and right side of the lower jaw is assigned '4'. Front teeth are assigned position number '1' in each segment, and numbering proceeds through '8' which is assigned to the wisdom tooth in each segment. The figure shows the extracted codes based on restored surfaces as well as the corresponding dental chart.

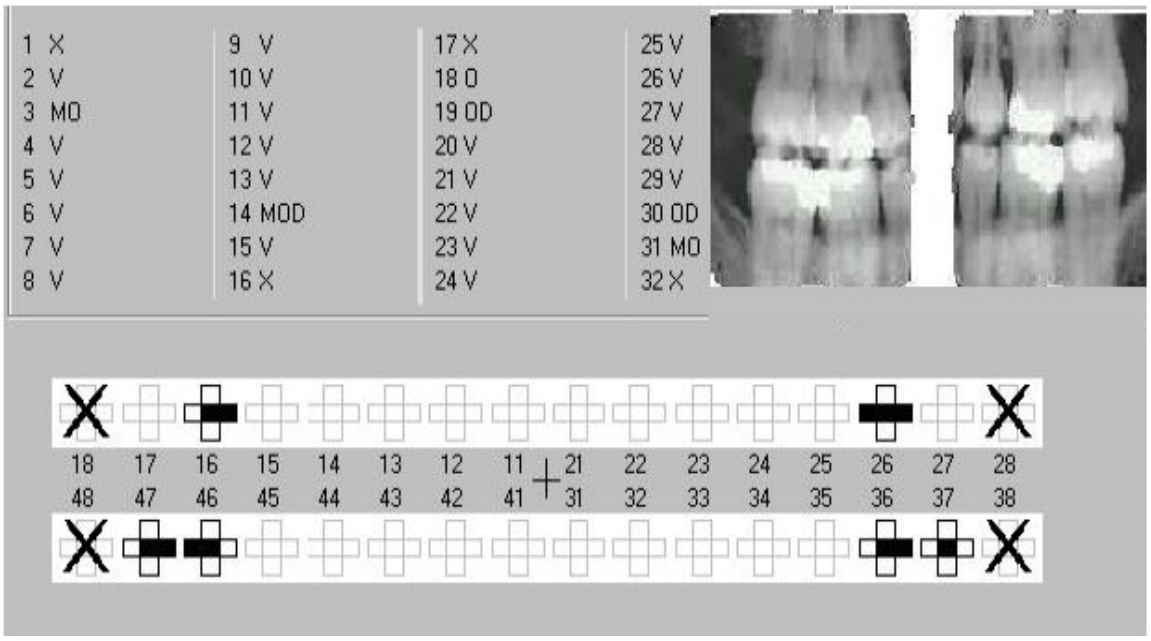


Figure 2.1: An example of WinID encoding and charting.

2.1.2 Java 2 Platform, Enterprise Edition (J2EE): [J2ee]

The development of J2EE applications requires a multitiered approach to application development. Unlike the fat-client approach in which a single application included presentation logic, business logic, and data access (resources) logic, the distributed application takes this functionality and places it in various components across multiple logical tiers.

A simple version of multitiered architecture involves three logical tiers: client tier, business tier, and resource tier. The client tier manages the client interface and communicates with the business tier to obtain the information needed to present to the

user. The business tier reacts to client calls and retrieves data from the resource tier as needed.

The three important tiers are explained below:

- **Presentation Tier:** The presentation tier is responsible for the preparation of the output to the client tier. Since in a Web application we are usually preparing these pages dynamically, this tier must be able to store and retain information between calls, either in memory or in a data store.

  If we are using a Web browser as our client, then the protocol between the client tier and the presentation tier is Hypertext Transport Protocol (HTTP). The most logical server for the presentation tier is a server that can perform HTTP, such as Apache. Additionally, we would like the server to be able to manage dynamic content using a robust language such as Java. The Tomcat server, the reference implementation of the Java servlet engine, provides this capability using servlets and JSP pages.

  The components used most often on this tier are either Java servlets, JSP pages, JavaBeans, and tag libraries.

- **Business Tier:** The business tier isolates, encapsulates and codifies the business logic (the business rules) of the organization in the application. These are rules

such as how to select sales regions for sales reports, including the usual list of exceptions that always seem to exist for many business rules.

Logic that could have resided in the presentation tier in Web components such as JSP pages or servlets is effectively pushed off and managed in this tier. Here in the business tier, under their auspices of an application server, the component is easy to look up and use, thus enhancing reusability.

The components on this tier can be created using a variety of Java technologies: Java Beans, tag libraries, and Enterprise Java Beans (EJB), or remote components delivered using RMI.

- **Resource Tier:** The resource tier contains a number of systems and services that tend to be shared resources within an application. Such resources can include databases, message queues, and brokers.

2.1.3 JavaServer Pages (JSP): [Java2] [Jsp] [Scwcd]

The JavaServer Pages (JSP) technology provides a simplified, fast way to create web pages that display dynamically-generated content. JSP technology was designed to make it easier and faster to build web-based applications that work with a wide variety of web servers, application servers, browsers and development tools. JSP unlike ASP (Active

Server Pages) work on any web or application server, helps in separating the application logic from the appearance of the page, allows fast development and testing and simplifies the process of developing interactive web based applications.

The Syntax elements prominently used in JSP are as follows:

- **JSP Directives:** JSP pages use JSP directives to pass instructions to the JSP engine. A JSP Directive starts with <%@ and ends with %>.The different directives used are as follows:
    1) Page Directive: This directive informs the JSP engine about the various properties of a JSP page like buffer information, thread information and error handling.
    2) Include Directive: This directive is used to include contents of external file in the JSP page.
    3) Taglib Directive: This directive indicates a library of custom tags that the page can invoke.
- **JSP Action Tags:** These tags are the standard core tags used by the JSP engine. The syntax of the action tags are <jsp:actiontag> The different action tags are as follows:
    1) jsp:useBean: This tag declares the usage of an instance of a JavaBeans component. If the Bean does not already exist, then the JavaBean component instantiates and registers the tag.
    2) jsp:setProperty: This sets the value of a property in a Bean

3) jsp:getProperty: This tag gets the value of a Bean instance property, converts it to a string, and puts it in the implicit object "out".
4) jsp:include: This tag is used to include the contents another JSP page in the current JSP page.
5) jsp:forward: This tag is used to forward the command from present JSP page to another JSP page.

- **JSP Declaration:** The Declaration is used to initialize any variables in the JSP page. The declaration is a JSP page starts with a <%! and ends with %>.
- **JSP Scriplet:** This is any executable java code put between the tags <% and %> in a JSP page.
- **JSP Expression:** This is used to automatically print what ever is put between the tags <%= and %>.
- **JSP Comment:** This is used to put any comment in the JSP page. A JSP comment starts with <%-- and ends with --%>.

2.1.4 Java DataBase Connectivity (JDBC): [Stan] [Java2]

Call-level interfaces such as JDBC are programming interfaces allowing external access to SQL database manipulation and update commands. They allow the integration of SQL calls into a general programming environment by providing library routines which interface with the database. In particular, Java based JDBC has a rich collection of routines which make such an interface extremely simple and intuitive.

JDBC technology is an API that provides cross-DBMS connectivity to a wide range of SQL databases and access to other tabular data sources, such as spreadsheets or flat files. If a java program has to interact with the database, a connection can be opened to the database using the standard library routines. JDBC is used to send the SQL code to the database and process the results that are returned. The connection is closed when the program is done working on the database.

JDBC provides a set of high-level classes that enable anyone acquainted with SQL and Java to write database applications. Considerations like networking and database protocols are transparent to the application programmer. These are handled by classes within JDBC drivers.

Oracle provides two main types of drivers.

The OCI Driver:

The OCI (type 2) driver consists of java wrappers to the low-level Oracle Call Interface (OCI) libraries used by utilities like SQL*Plus to access the database server. The OCI driver offers potentially better performance than the thin driver. It however requires the OCI libraries to be installed on the local machine.

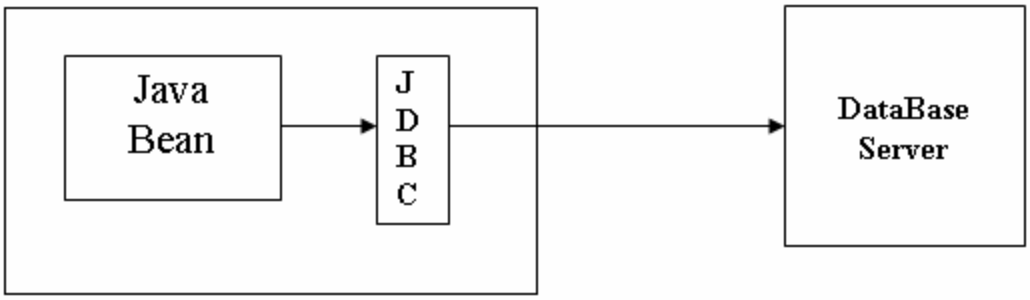
The "thin" Driver:

Also referred to as type 4 driver, the thin driver is a pure Java implementation of Oracle's networking protocol (Net8). Being self-contained, it may be used on any machine with or without Oracle installed or even distributed with application classes in an applet.

2.1.5 JavaBeans: [Oreilly] [Java] [Bean-Spec]

'Components' are self-contained elements of software that can be controlled dynamically and assembled to form applications. Apart from this these components must also interoperate according to a set of rules and guidelines. JavaBeans is Java's component model. It allows users to construct applications by piecing components together using programmatically.

A Java Bean is a reusable software component that is written in Java programming language which supports the features of software reuse, component models and object orientation. Java Beans can be simple GUI elements such as buttons and sliders or can be visual components such as database viewers.



Java Application

Figure 2.2: Block Diagram for Java Database Connectivity from Java Beans.

Individual Java Beans will vary in the functionality they support, but the typical unifying features that distinguish a Java Bean are:

- Support for "introspection" so that a builder tool can analyze how a bean works
- Support for "customization" so that when using an application builder a user can customize the appearance and behaviour of a bean.
- Support for "events" as a simple communication metaphor than can be used to connect up beans.
- Support for "properties", both for customization and for programmatic use.
- Support for persistence, so that a bean can be customized in an application builder and then have its customized state saved away and reloaded later.

The three most important features of a Java Bean are the set of properties it exposes, the set of methods it allows other components to call and the set of events it fires. 'Properties' are named attributes associated with a bean that can be read or written by calling appropriate methods on the bean. A property 'abc' that represents a string can be read by calling a "String getAbc()" method and updated by calling a "void setAbc(String a)" method. 'Methods' in a Java Bean are just normal Java methods which can be called from other components or from a scripting environment. 'Events' provide a way for one component to notify other components that something worth noticing has happened. An event listener object can be registered with an event score that detects that event has

occurred and calls the appropriate method on the event listener object.

2.1.6 Java Native Interface (JNI): [Java]

The Java Native Interface (JNI) is the native programming interface for Java. JNI helps our code to be completely portable across all platforms. The JNI allows Java code that runs within a Java Virtual Machine (VM) to operate with applications and libraries written in other languages, such as C, C++ and assembly. The situations in which JNI native methods and JNI is used are as follows:

- The standard Java class library may not support the platform-dependent features needed by our application.
- We may already have a library or application written in another programming language and we wish to make it accessible to Java applications.
- We may want to implement a small portion of time-critical code in a lower-level programming language, such as assembly, and then have our Java application call these functions.

JNI serves as a glue between Java and native applications. The following diagram shows how the JNI ties the C side of an application to the Java side.

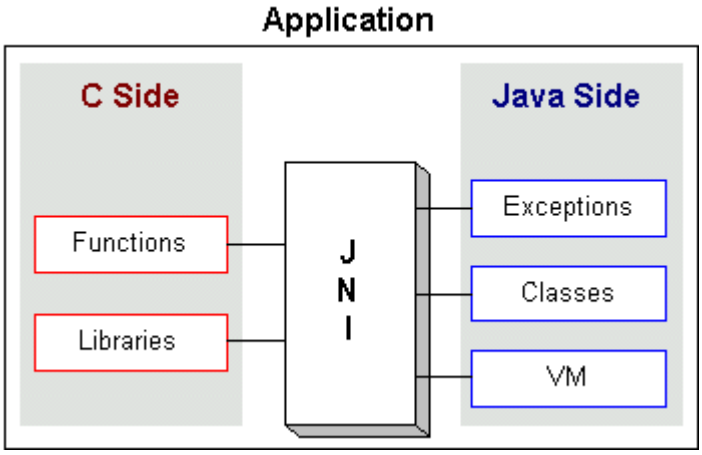


Figure 2.3: Block Diagram for JNI acting as glue between C and Java. [Java]

2.1.7 JavaScript: [Net] [Java3]

JavaScript is an object-based scripting language for client and server applications. JavaScript and Java are similar in some ways but fundamentally different in others. The JavaScript language resembles Java but does not have Java's static typing and strong type checking. JavaScript supports most Java expression syntax and basic control-flow constructs. In contrast to Java's compile-time system of classes built by declarations, JavaScript supports a runtime system based on a small number of data types representing numeric, Boolean, and string values. JavaScript has a simple, instance-based object model that still provides significant capabilities. JavaScript also supports functions without any special declarative requirements. Functions can be properties of objects, executing as loosely typed methods.

Java Scripts can be used for the following kinds of solutions:

- Make the web page to respond or react directly to user interaction with form elements like input fields, test areas, buttons, radio buttons, check boxes and hyper text links.
- Distribute small collections of database-like information and provide a friendly interface to that data.
- Control multiple-frame navigation, plug-ins, or Java applets and on user choices in the html document.
- Want the data to be processed on the client before submission to a server.

2.2 Database and Structured Query Language (SQL):

SQL is an ANSI (American National Standards Institute) a standard computer language for accessing and manipulating databases. SQL statements are used to retrieve and update data in databases like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase etc. Oracle is a preferred database for industrial use when compared to others for its award winning features. Oracle is a fourth generation relation database management system and has the ability to reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the same data. This is also secure from unauthorized access and provides efficient solutions for failure recovery.

The major keywords of SQL used in almost all the different versions of SQL language

are as given below:

- The commands of Data Manipulation Language (DML) part of SQL that is used to get or insert the data into the tables already created are:
    1) SELECT extracts data from a database table.
    2) UPDATE updates data in a database table.
    3) DELETE deletes data from a database table.
    4) INSERT INTO inserts new data into a database table.
- The commands in Data Definition Language (DDL) part of SQL which permits database tables to be created or deleted are as follows:
    1) CREATE TABLE creates a new database table.
    2) ALTER TABLE alters or changes a database table.
    3) DROP TABLE deletes a database table.
    4) CREATE INDEX creates an index
    5) DROP INDEX deletes an index.
- The DISTINCT keyword is used to return only distinct values.
- The WHERE clause is used to conditionally select data from a database table.
- The operators used with WHERE clause to specify a selection are = (equal), <> (not equal), > (greater than), < (less than), >= (greater than or equal), <= (less than or equal), BETWEEN (between an inclusive range), LIKE (search for a pattern).

2.2.1 Constraints:

Constraints are declarations of conditions about the database that must remain true. The system checks for violation of the constraints on actions that many cause a violation, and aborts the action accordingly. There are five integrity constraints in Oracle.

- **Not Null:**

  A column in a table can be specified that I can not take values which are null. It's not possible to insert a null in such a column. The default is null.

  Syntax:

  Create table test (a number not null, b null);

- **Unique Key:**

  The unique constraint doesn't allow duplicate values in a column.

  Syntax:

  Create table test (a number unique, b number);

- **Primary Key:**

  A primary key combines a unique and a not null constraint. A table can have at most one primary key. After creating a primary key, it can be referenced by a foreign key.

  Syntax:

  create table test (a number primary key, b);

- **Foreign Key:**

  A foreign key on a column ensures that the value in that column is found in the

primary key of another table.

Syntax:

create table test (a number references test1(Id), b number);

- **Check:** A check constraint allows to state a minimum requirement for the value in a column.

  Syntax:

  create table test (a number check (a between 0 and 100), b number);

CHAPTER 3

PROBLEM STATEMENT AND RESEARCH OBJECTIVES

3.1 Problem statement:

A web application is an application delivered to users from a web server over a network such as World Wide Web or an intranet. Web applications are popular due to the ubiquity of the web browser and have the ability to update and maintain web applications without disturbing and installing software on potentially thousands of client computers. Though many variations are possible, a web application is commonly structured as a three-tiered application. In this common form, a web browser is the first tier, an engine using some dynamic web content technology (e.g. Java Servlets) is the middle tier, and database is the third tier.

Although ADIS was previous implemented to get results on a local machine, there was always a scarcity in the form of a web interface that can be used by distant users interested in using Automated Dental Identification System built in WVU, MSU and UM. There was also a need for a database that can be used to store all the dental records and the data got by applying preprocessing on these dental records.

Problem:

Given a subject record, the web application should allow the user to upload the dental record onto the server after the user is authenticated. The server should invoke Matlab to do preprocessing steps like cropping, enhancement, segmentation, film classification on the dental record uploaded and then should run the potential match algorithm and should give back a candidate list of reference records from the database. The candidate list should be fed into the image comparison module to get back a final match list.

This application should also enable uploading of reference records on to the database. When the reference record is uploaded the preprocessing steps should be run on the record submitted and the post preprocessing data should be automatically uploaded onto the database.

This application should also be designed in such a way that the researchers from Michigan State University (MSU) and UM can also use the database designed in WVU for their research purposes.

3.2 Research Objectives:

The objectives of the research are summarized as below:

1) Design and implement a database to store the dental records and the preprocessing

dataof the dental record.

2) Develop a web application that takes in a subject record and interacts with the database with the reference records and give back the match list as result.
3) Develop bridge modules to make the interactions of the modules designed by other universities with the database designed in WVU possible.

CHAPTER 4

Web-ADIS and its Components

4.1 Introduction

In this chapter Web-ADIS and its components are discussed along with the role being played by the each of the component in Web-ADIS. The main components of Web-ADIS are discussed in section 4.1. The components of Web-ADIS are discussed in detail in sections 4.2, 4.3 and 4.4 respectively. The different technologies used in developing Web-ADIS are discussed in section 4.5. The different modules that play important roles in Web-ADIS and their functioning are discussed in section 4.6.

4.2 Components in Web-ADIS:

The design of Web-ADIS can be classified mainly into three main components:

- Web-ADIS client
- Web-ADIS server
- Database server

The tree diagram for Web-ADIS is as given below:

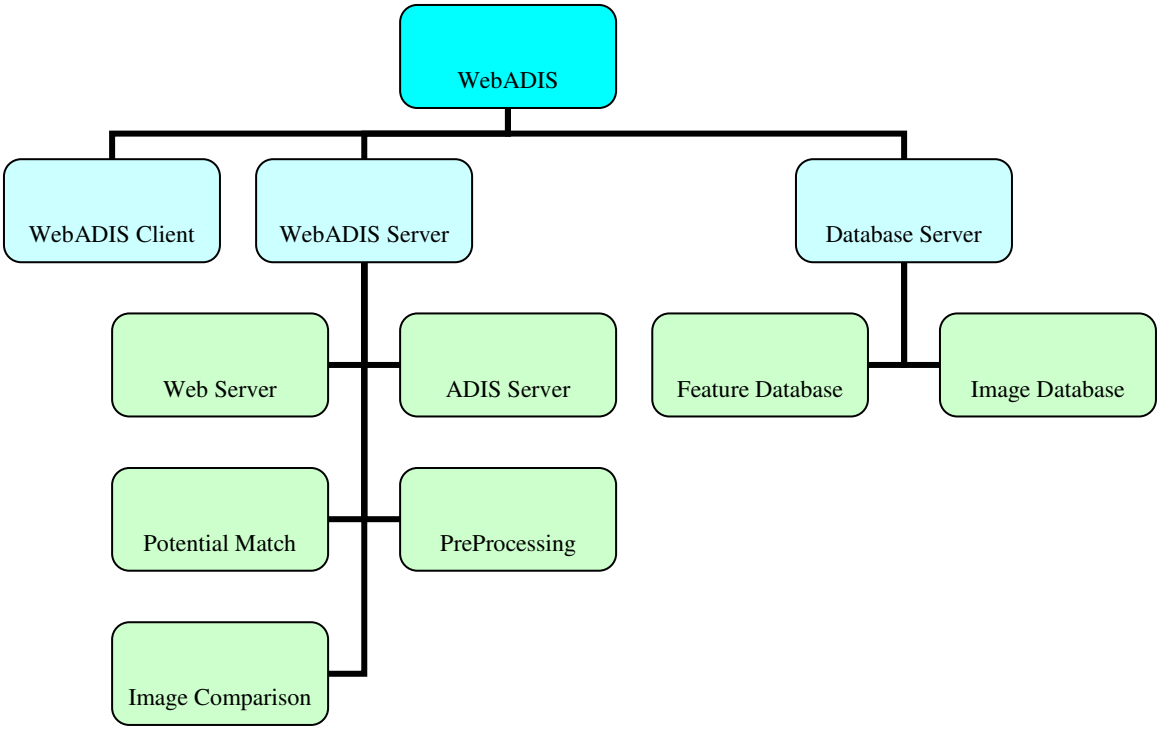


Figure 4.1: Tree Diagram of Web-ADIS

Web-ADIS Server and Database Server are a combination of five and two sub-components respectively. The subcomponents of Web-ADIS Server are Web Server, ADIS server, Potential Match Server, Preprocessing Server and Image Comparison Server. The sub-components of Database Server are Feature Database and Image Database.

4.3 Web-ADIS Client:

The Web-ADIS client can be technically defined as a user using valid browser software and is trying to connect to Web-ADIS server. An authenticated user can upload a subject record into Web-ADIS server and can get back the match list.

An authenticated client if authorized can also update the Web-ADIS server with new algorithms which will help the system to be more effective in terms of speed of response and accuracy. It also helps to keep the system upto date with newly developed algorithms which are proven to give better results than the presently used ones.

4.4 Web-ADIS Server:

This component plays the role of a controller for all the work being done by Web-ADIS. This commands the flow of signals in the system. When a request is sent from Web-client, the processing of the request is done by Web-ADIS server by communicating with the database server. The results are returned to the client when the p rocessing of request is complete.

A brief description of sub-components of Web-ADIS server is given below:

4.4.1 Web Server:

The main job of Web server is to securely connect Web-ADIS server to outside world

through internet. Web server can be defined as software responsible for serving Web pages via the http protocol to clients, mostly using browser [En]. Web server has a unique address so that other computers connected to internet can identify it on the vast network.

4.4.2 ADIS Server:

ADIS Server acts as a controller for Web-ADIS and controls the flow of signals to Preprocessing Server, Potential Match Server, Image Comparison Server and Database Server. When a request is made by the user the ADIS Server first evaluates the type of request depending on the mode of operation, prompts the other servers to act on the request and when the result is ready to be send back to the user, it prompts the Web Server to display the result to the user.

4.4.3 Preprocessing Server:

The Preprocessing server is responsible for the preprocessing of the dental record which is submitted by the user before it passes on to Potential Match Server and Image Comparison Server. The Objective of Preprocessing stage is to process the record to get it ready for comparison to the reference records in the database. The various processing steps are as discussed below:

4.4.3.1 Cropping:

A dental record is a collection of many dental films. Cropping can be defined as a process of identifying the different films from the record. Figure 1-2 shows a dental record which returns films after passing through cropping module.

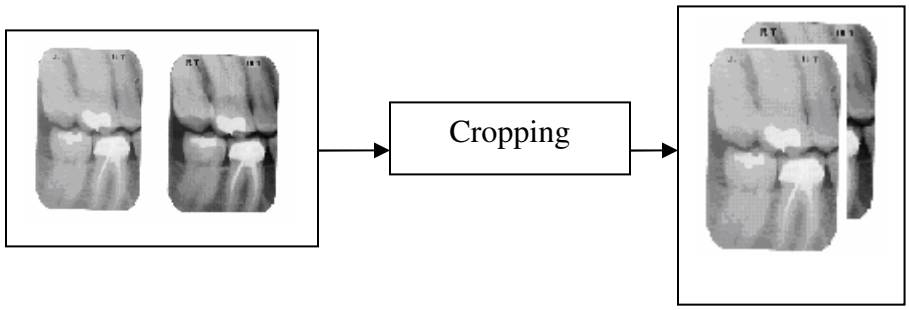


Figure 4.2: A record passes through Cropping module to get a cropped films.

4.4.3.2 Enhancement:

In general a superior quality image is easier to preprocess and match while a poorer quality image has more noise and harder to segment. Often the images in DIR or the subject image may be poor in quality as a result making them harder to segment [Ikkry04]. Enhancement improves the area of teeth in the image by reducing the noise against the background.

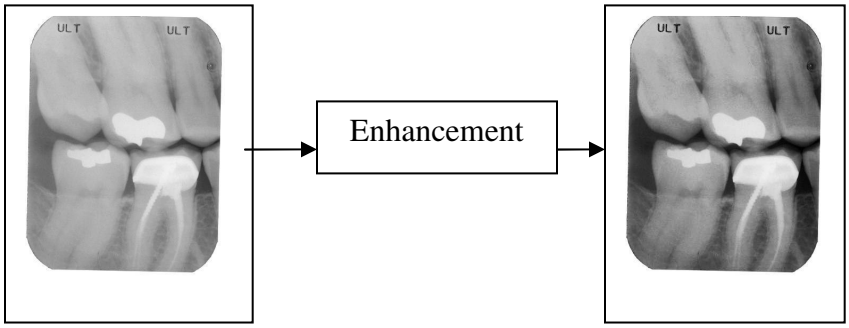


Figure 4.2: A film passes through Enhancement module to get a enhanced image.

4.4.3.3 Segmentation:

The realizations in Image Comparison Component are based on comparison of individual teeth. Segmentation partitions an image into its constituent regions and extracts the objects of interest and this plays a critical role in most subsequent image analysis especially in pattern recognition and image matching. Each segmented tooth represents a Region of Interest (ROI) that contains distinctive features used in the subsequent steps of identification. We define a qualified ROI of as a rectangular area in the image of a dental film that bounds one tooth [Eyad]. Figure 4.3 shows the segments which are identified by the module when a film is sent as input.

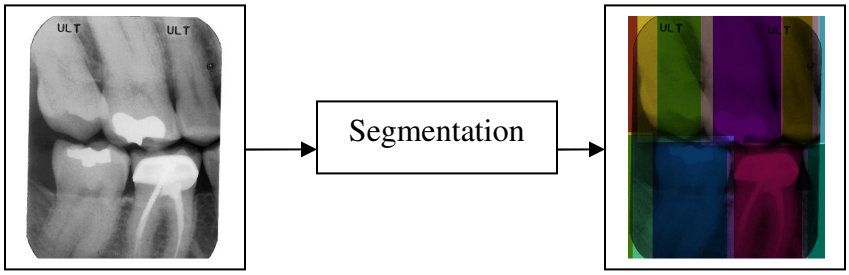


Figure 4.3: A film passes through segmentation module to get a segmented image.

4.4.3.4 Film Classification:

The film is a digitized x-ray of a section of mouth. There exist several types of films and they are categorized based on the section of the mouth they capture. The films can be categorized into three categories: Bitewing, Periapical and Panoramic views. A bitewing is a view of both upper and lower jaw in a single film, a bite as shown in figure 4.4. A Periapical as shown in figure 4.5 shows only one jaw, either an upper or lower jaw but not both. A Panoramic shows a panoramic view of all the teeth on upper and lower jaw. Panoramic view can be seen in figure 4.6.

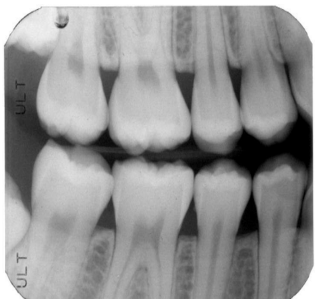


Figure 4.4: Bitewing view

Figure 4.5: Periapical view

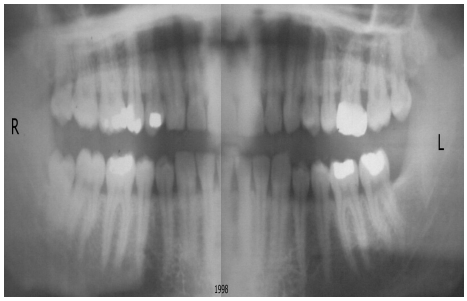


Figure 4.6: Panoramic view.

4.4.3.5 Contour Extraction:

Contour Extraction module extracts the border of the tooth in the film. The region of interest for the image comparison is exact tooth region in the film, if done can give better results. Efficient contour extraction technique can differentiate between region of interest in a tooth to background and region of neighboring tooth as shown in Figure 4.7.

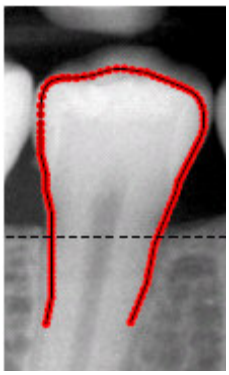


Figure 4.7: Contour Extraction

4.4.3.6 Teeth Labeling:

Image comparison involves comparison of a type of teeth of a subject record to its respective counterpart in the reference record. A shown in the figure 4.8 the labeling module labels each tooth to a number according to the section of the mouth it is present in. The comparison module while working on a tooth in a subject film looks out for a corresponding tooth in the reference record with the same label for comparison.

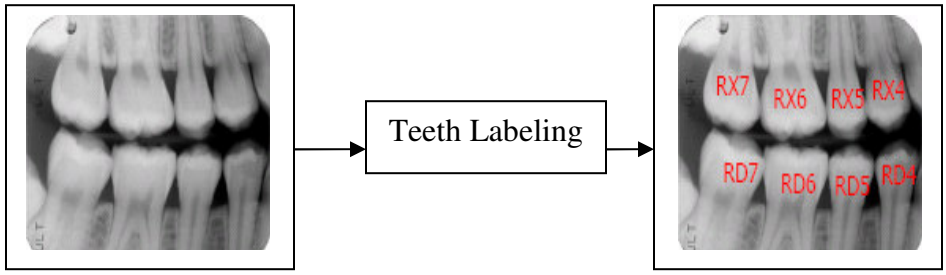


Figure 4.8: A film after passes through the Teeth Labeling module to get a labelled image.

4.4.4 Potential Match Server:

The subject record should be compared to images in the reference database to get a match. As the reference database has millions of records the realizations that are chosen to run on the reference records should have an initial motive to filter the list of

possibilities of match to a minimal number. Potential Match Server works on the reference records with a motive to bring down the search list from millions of records to few thousands. The records for the list are selected based on the resemblance of dental/non-dental features with the subject record. Dental features may comprise of shape, color and texture while non-dental features may have age, gender etc. The final list is called Candidate List. The algorithms used in the Potential Match Server are less effective than that used in Image Comparison Server.

The Potential Match algorithm used in ADIS is based on Image Signature service provided by Oracle. Oracle interMedia describes the ORDImageSignature object type, which supports content-based retrieval (image matching). Signature of the image contains color, texture and shape information of the image and is stored in a BLOB. The ORDImageSignature operators used here are IMGSimilar() and IMGScore( ). IMGSimilar() determines whether or not two images match. Specifically, the operator compares the signature of a query image with the signatures of images stored in a table, and determines whether or not the images match, based on the weights and threshold value. This operator returns 1 if the computed distance measure (weighted average) is less than or equal to the threshold value (indicating a match), and returns 0 when the distance between the two images is more than the threshold value. The attributes for IMGSimilar() involve two ORDImagesignatures of the image to be compared, weights and thershold. A string consisting of matching attribute names followed by values between 0.0 and 1.0. The matching attributes refer to the weights assigned by the user to

the different attributes that influence the kind of match. Attributes not specified by the user have a default value of 0.0.

The string can have all or some of the attributes. The value associated with an attribute specifies its relative importance in determining the distance between the signatures. An attribute with a value of 0.0 is not considered at all, while an attribute with a value of 1.0 is of the highest importance. The weights supplied are normalized prior to processing such that they add up to 1.0, maintaining the ratios that you supplied. At least one of the attributes must have a value greater than 0.0. The attributes are the following:

- color: A value between 0.0 and 1.0 indicating the importance of the feature color.
- texture: A value between 0.0 and 1.0 indicating the importance of the feature texture.
- shape: A value between 0.0 and 1.0 indicating the importance of the feature shape.
- location: A value between 0.0 and 1.0 indicating the importance of the location of the regions in the image. The location weight string cannot be specified alone, it must be used with another weight string.

The threshold value is with which the weighted sum of the distances is to be compared. If the weighted sum is less than or equal to the threshold value, the images are considered to

match. The range of this parameter is from 0.0 to 100.0.

IMGScore() compares the signatures of two images and returns a number representing the weighted sum of the distances for the visual attributes. This function returns a float value between 0.0 and 100.0, where 0.0 means the images are identical and 100.0 means the images are completely different.

Let N_{Sub} be the number of films in subject record and N_{Ref} be the number of films in each reference record in the database. For two parameters e and p let Sub be defined as

$Sub = (1-e)^p$

Let Tail, Root and Distance be some variables.

The Potential Match Algorithm can be evaluated into two cases based on number of subject and reference records:

1) $N_{Sub} \geq N_{Ref}$

For this case e = .5 and p = 2.

For Each Film in reference record

Score = score for best match when compared to all films in subject record.

Total = Total + $(1- Score)^2$

Tail = Absolute (N_{Sub} - N_{Ref})

Root = Total + Tail * Sub

Distance = √Root

2) $N_{Sub} < N_{Ref}$

For this case Sub = .25.

For Each Film in subject record

Score = score for best match when compared to all films in reference record.

Total = Total + $(1- Score)^2$

Tail = Absolute ($N_{Sub} - N_{Ref}$)

Root = Total + Tail * Sub

Distance = √Root

A (subject record, reference record) pair with least distance is considered the best matching pair. Dissimilarity between the pair increases as the distance increases.

4.4.5 Image Comparison Server [Nassar 03]

The candidate list is compared with subject record on one-to-one basis and a probability of match of subject record to reference record in candidate list and a decision whether subject record is a match to the reference record in candidate list in given as output. A threshold is set on probability of match to filter the candidate list to a refined one called

Match List. The number of record in match list is ideally one and practically a minimal number greater than one.

Image Comparison consists of two stages:

- Pre-Processing
- Decision Making.

Pre-Processing:

The inputs here are subject and reference records from candidate list. A series of steps like noise reduction, enhancement, segmentation, alignment and compression are run on input record to smoothen the process of decision making.

Decision Making:

The decision making stage extends low-level features from compressed region of interest pairs and measures the difference between respective features and finally uses the measured differences to come up with an estimate of the matching probability between the submitted pair.

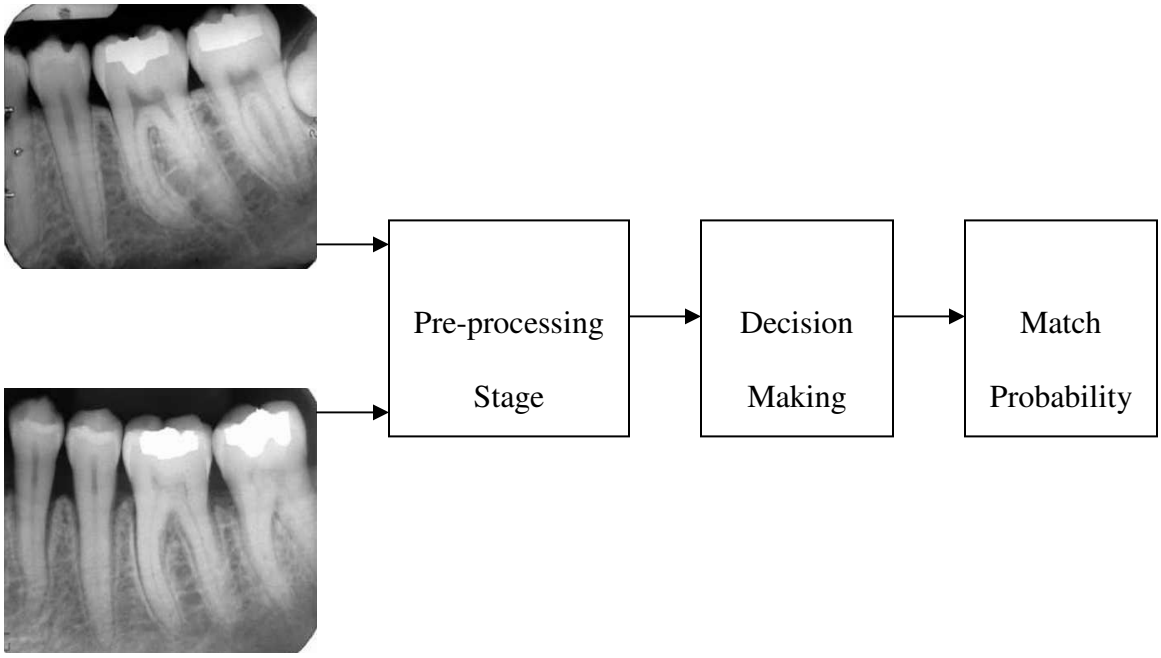


Figure 4.9: Block Diagram for Image Comparison Diagram

The match list if has more than one dental record is sent to forensic experts who examines the dental records in match list and the subject record to get the exact match.

4.5 Database Server:

The subject record submitted during identification mode should be compared to the dental/non-dental features of the reference images to find the final match. These features of reference images and the dental images of reference records are stored in the database so that they can be accessed while caring out identification process on the subject record. For the ease of accessing and storage of information the database server is divided into

two parts. The sub-components in the database server are as follows:

4.5.1 Image Database:

All the reference dental images are stored in image database also called Digital Image Repository (DIR). This is also the computerized repository to all users authorized for maintenance mode usually users from the Criminal Justice Information Services (CJIS) to update the references images to the database. The users trying to identify a subject in the identification mode can access the images, which are used by the Potential Match Server and the Image comparison Server to compare with the image of the subject record in process to get the match.

4.5.2 Feature Database:

This is the store house for the dental and non-dental features of all the reference dental images. Dental features are the features extracted from the reference images after the preprocessing steps are run on the reference dental images. Non-dental features comprises of features that are not related to dental images, these comprise of details like name, age, gender etc. The Feature Database can be updated by authorized users and can be accessed by the users trying to identify a subject record.

4.6 Implementation of Web-ADIS:

ADIS Web interface is developed using J2EE. J2EE technology was used because it's the most secure and Efficient Technology. The Web server used for hosting WebADIS is Apache Tomcat. The other technologies that were considered before zeroing in on J2EE and Tomcat are Dotnet technology which requires servers running on MS Windows NT or other Microsoft products. It is important to note that the Microsoft Products are not secure on the server side and are not as efficient as others like the Apache Tomcat Server. These were the main reasons behind using J2EE technology which uses the Apache Tomcat 5.5 to process the user requests. The Tomcat server is a Java based Web Application container that was created to run Servlets and Java Server Pages (JSP) in Web applications. As part of Apache's open source Jakarta project, it has nearly become the industry accepted standard reference implementation for both the Servlets and JSP API. Another reason behind choosing JSP for the Web interface was that the basic architecture of the Web-ADIS is Object Oriented and secondly it is relatively easy to interface JSP with java (through JavaBeans) which is providing interface between different components. Java is also used to interface the Web Server with the ADIS server. It is important to note that the ADIS Server also uses JAVA to communicate with other Servers which includes Potential Match Server, Preprocessing Server and Image Comparison Server. Although full fledged Matlab servers are not being used for Preprocessing and Image Comparison, Matlab engine is invoked using JNI (Java Native Interface). The JNI is

for programmers who must take advantage of platform-specific functionality outside of the Java Virtual Machine. The JNI allows Java code that runs within a Java Virtual Machine (VM) to operate with applications and libraries written in other languages, such as C, C++ and assembly. In addition, the Invocation API allows you to embed the Java Virtual Machine into native applications.

JNI is used by programmers to write native methods to handle those situations when an application cannot be written entirely in the Java programming language. In this case Matlab supports only C and Matlab had to be called from java, so JNI is used to call the dynamic link library file created from the C program written to invoke the Matlab engine and do the preprocessing and image comparison.

The User Authentication is the integral part of ADIS Web interface. The users are asked to enter their Login ID and Password to access the Secured Web Interface. All the user information including user name, password and personal information is saved on the Dbase Server. We used Oracle 10 g. for our data base system for its award winning efficiency and security. All the information from the Web browser to the Dbase server is sent by using JDBC (Java Data Base Connectivity) and Java Beans.
ADIS Server uses JDBC, Java Native Interface (JNI), SQL, C/C++ and Java Beans Technology to handle all the requests and responses from both the users and the servers and uses JSP and JavaScript to present the results.

4.7 Working of WebADIS:

Apart from identifying the subject record which was uploaded by the user by comparing it with the reference records already present in the database, WebADIS also provides room for maintenance of system and configuring the system. The present version of WebADIS only supports identification mode and the maintenance mode. The working of the WebADIS for the two modules will be explained here:

4.7.1 Working of WebADIS in Identification Mode:

When an user wants to identify a subject record, he uploads the record on to the system and gets back all the reference records in the database whose dental/non-dental features match closely with subject record. The subject record has to pass through authentication module, Identification module, Preprocessing module, Potential Match module and Image Comparison module before the match list is obtained.

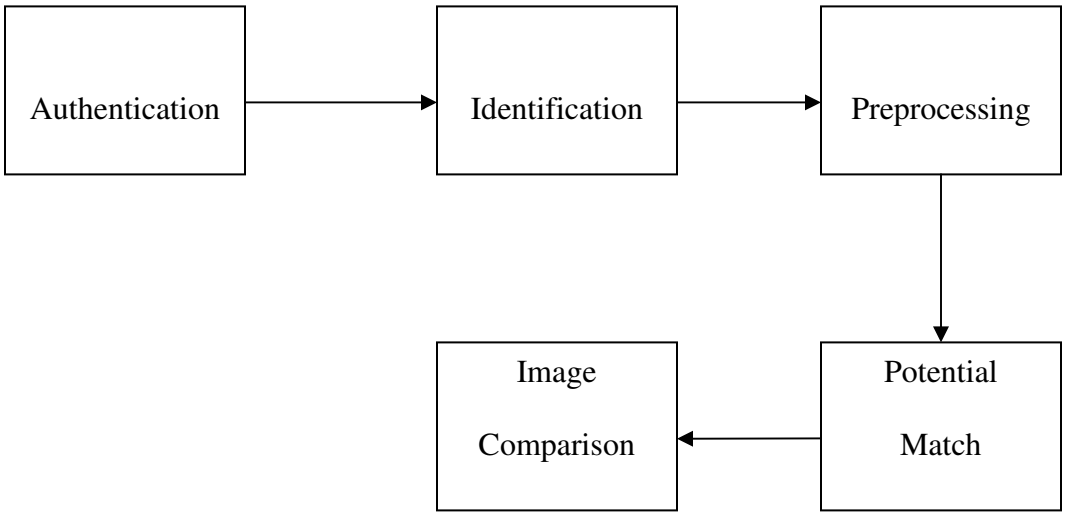


Figure 4.10: Block Diagram for working of WebADIS in Identification mode

The modules in the identification mode are complex implementations in themselves. The different modules are explained below:

Authentication Module:

Authentication is any process by which a system verifies the identity of a user who wishes to access it [Mtech]. WebADIS is a secured Web application to be used only by authenticated users.The authentication method used here is the username/password. When a user first tries to access WebADIS he will be prompted to enter username and password. The username and password will be sent to ADIS Server through the Webserver which will check it with the user information stored in the database.

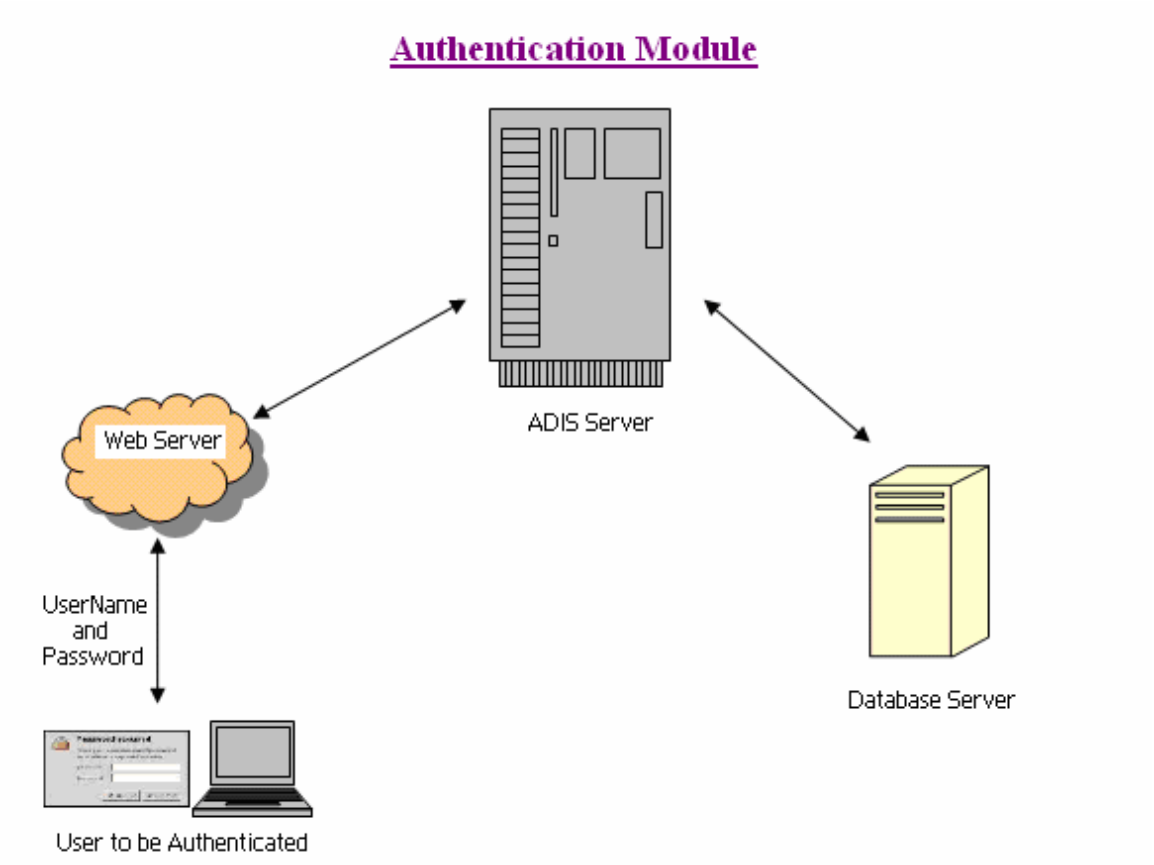


Figure 4.11: Block Diagram for Authentication Module

A person who is authenticated will be allowed to access the other modules in the Identification mode.

Identification Module:

An user after passing through the authenticated module, if authorized will be able to access the identification module. The identification module allows the user to select

the dental record and specify the mortality tag (AM or PM) of the dental record to be uploaded. The image gets uploaded into the local hard disk of the ADIS Server.

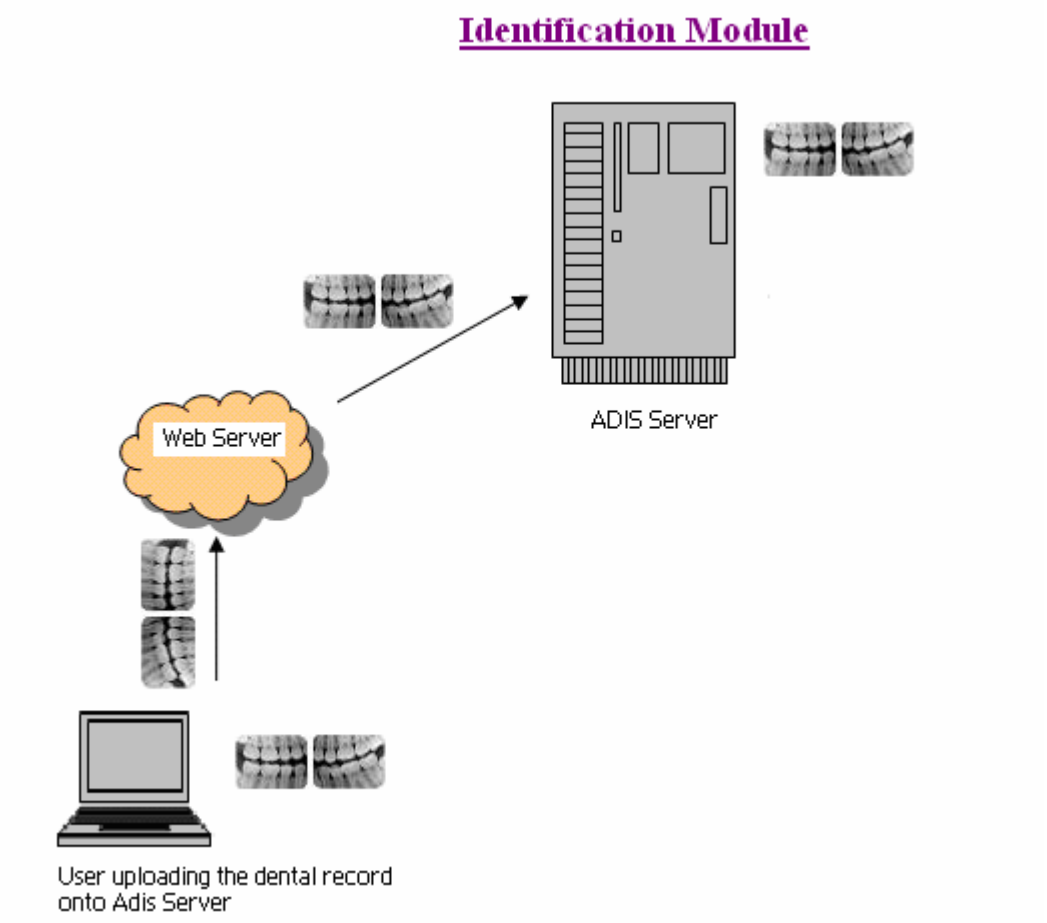


Figure 4.12: Block Diagram for Identification Module

Preprocessing Module:

The dental record uploaded on to the ADIS Server should pass through the preprocessing to get better match results. ADIS Server signals the Preprocessing Server to run the preprocessing algorithms on the dental record. Preprocessing

Servers runs the programs on the dental record to do cropping, enhancement, segmentation, film classification and labeling.

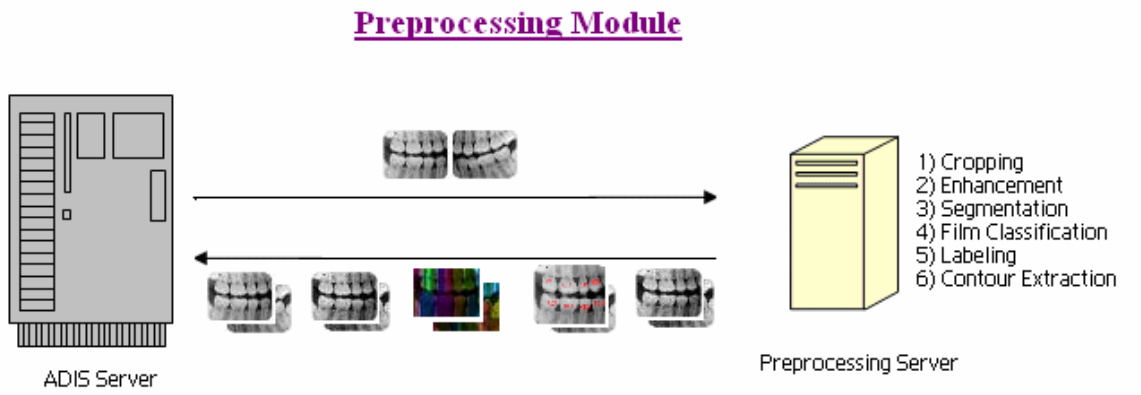


Figure 4.13: Block Diagram for Preprocessing Module

After the preprocessing is completed the command is given back to the ADIS Server.

Potential Match Module:

The processed dental records should be compared by potential match algorithms to the reference records to get back a candidate list. ADIS Server prompts the Potential Match Server and Database Server to run the potential match algorithms on the reference records in the database to get back a candidate list.

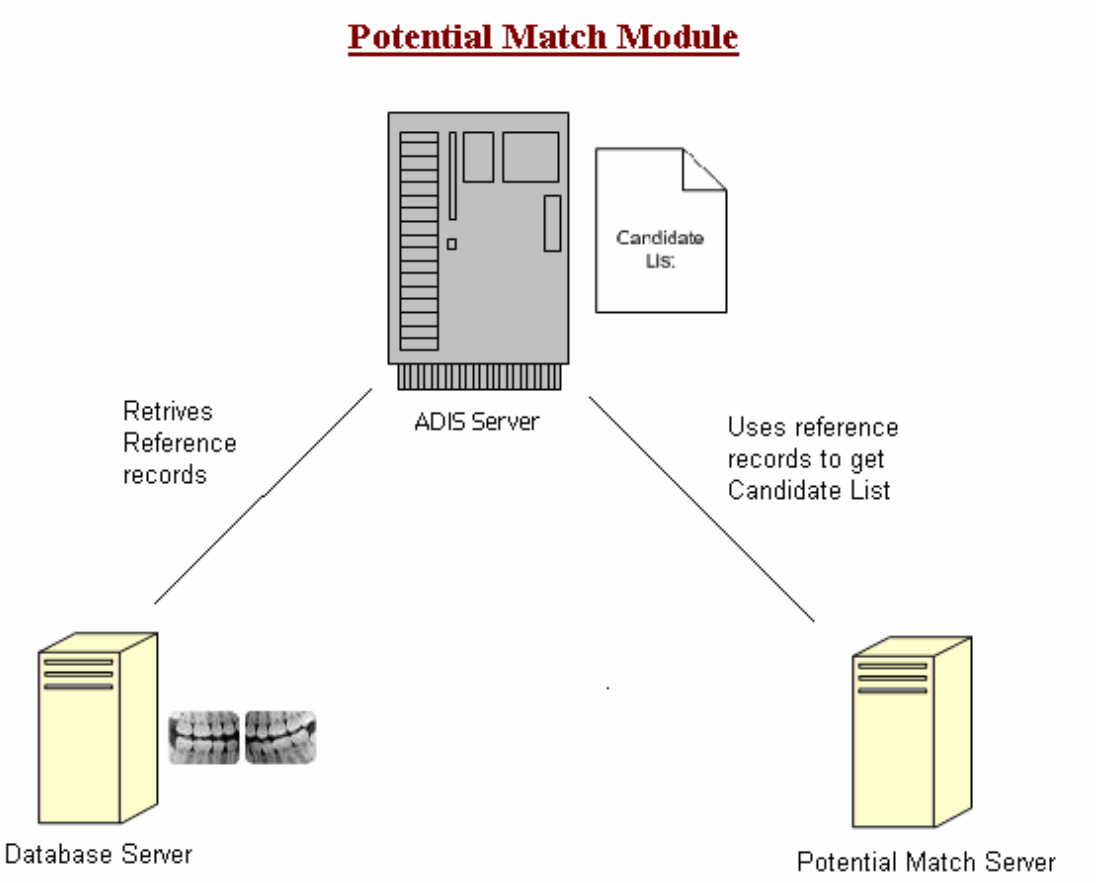


Figure 4.14: Block Diagram for Potential Match Module

Image Comparison Module:

The image comparison module considers only the records given in the Candidate List. It individually compares each reference record from the Candidate List to subject record and sorts out few records to give Match List. The number of records in the Match List is determined by the threshold.

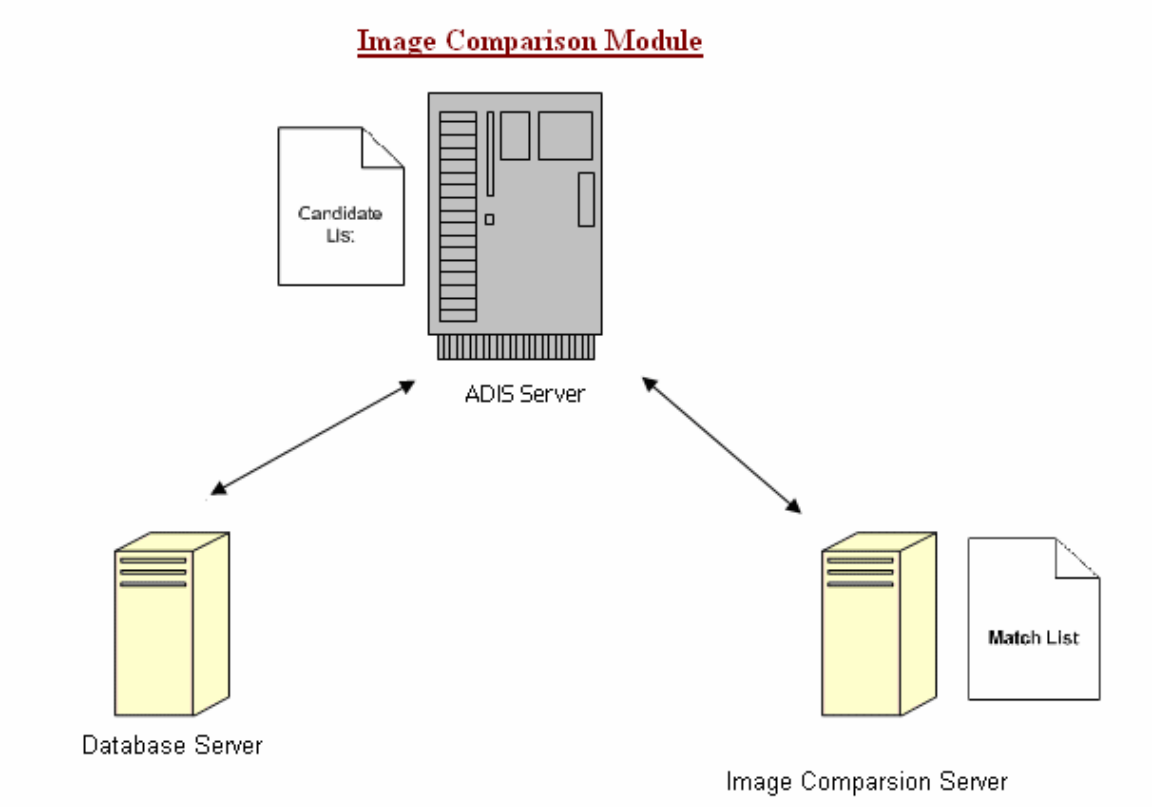


Figure 4.15: Block Diagram for Image Comparison Module

4.7.2 Working of WebADIS in Maintenance Mode:

A user who is authenticated should also be authorized to use the maintenance mode. The users who are given the authentication to use the maintenance mode are usually the officially appointed operators or the employees of the organization maintaining the database. The maintenance of WebADIS involves updating the algorithms of different realizations, updating the data in the records and uploading new reference

records on to the database. The present version of WebADIS supports only uploading of reference records in the maintenance mode. Maintenance mode comprises of Authentication module, Maintenance Module, Preprocessing Module and Database Module.

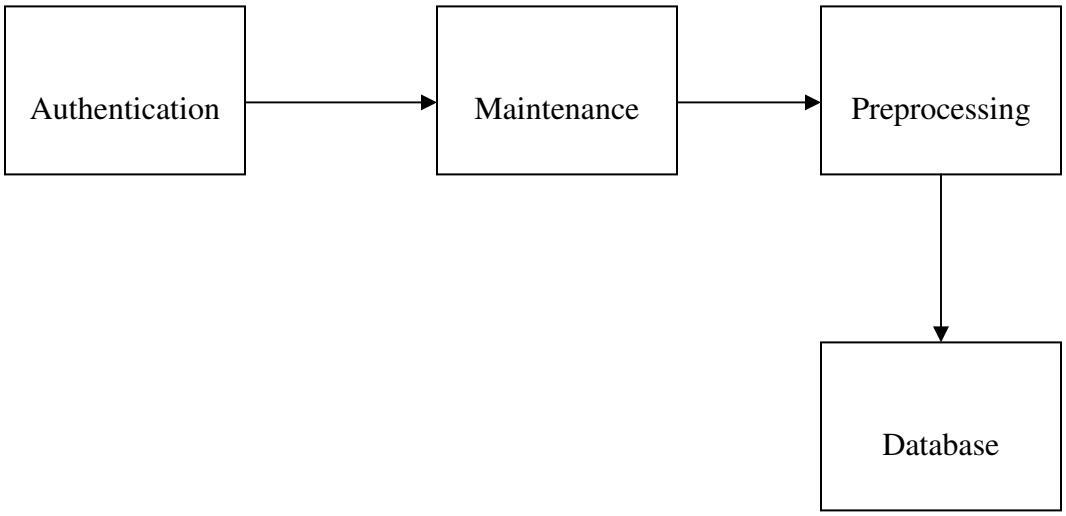


Figure 4.16: Block Diagram for working of WebADIS in Maintenance mode

Authentication Module:

This is same as that of the authentication module explained in the Identification Mode in section 1.6.1.

Maintenance Module:

Reference record should be accompanied by the information of the person whose dental

record is being uploaded. The information that need to be uploaded in the database are the mortality tag, name, gender, Date of birth, race, hair color, eye color, blood group, height and weight of the person whose dental record is being uploaded. These details might be of help in Potential Match stage to get the candidate list. The reference record is uploaded on to the ADIS Server from the local disk of the user who is uploading the record.

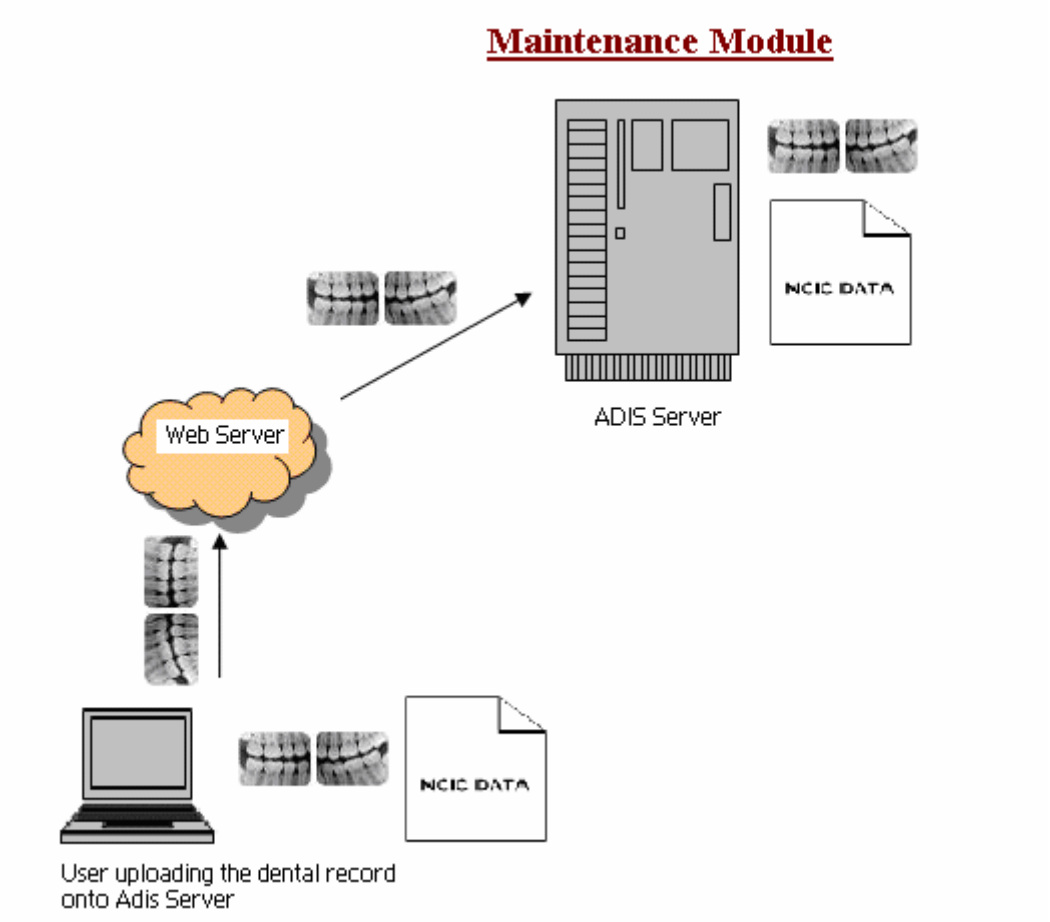


Figure 4.17: Block Diagram for Maintenance Module

Preprocessing Module:

The reference dental record uploaded onto the ADIS Server is preprocessed to get the preprocessing information before being uploaded on to DIR. The preprocessing information is fed on to the database for future use. The ADIS Server invokes the Preprocessing Server and passes on the record uploaded to do preprocessing on the record. The preprocessing steps done by the preprocessing server are explained in detail in section 4.4.3. The control is passed back to ADIS Server after the Preprocessing Server is done with its job.

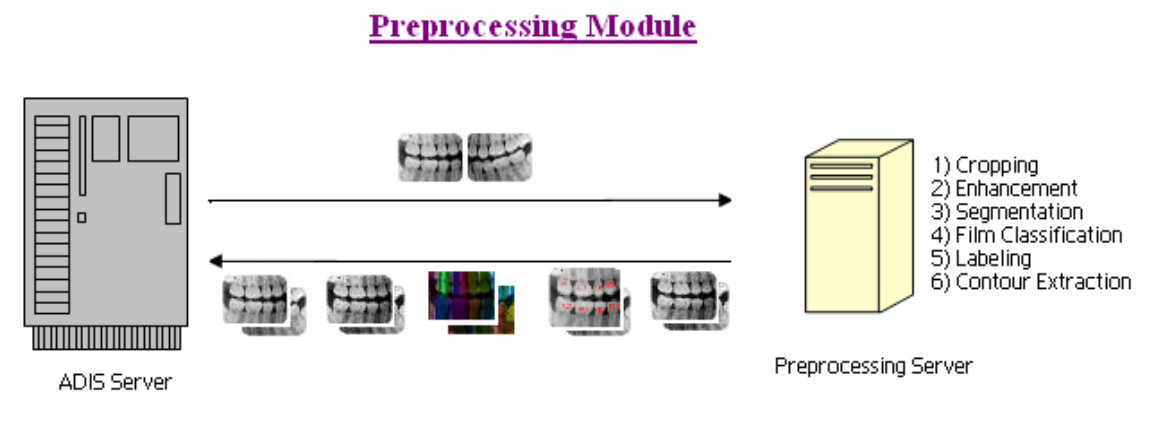


Figure 4.18: Block Diagram for Preprocessing Module

-

Database Module:

After the Preprocessing Server passes the command to ADIS Server, it signals the Database Server to store the dental record uploaded in DIR and the preprocessing data of the dental record.

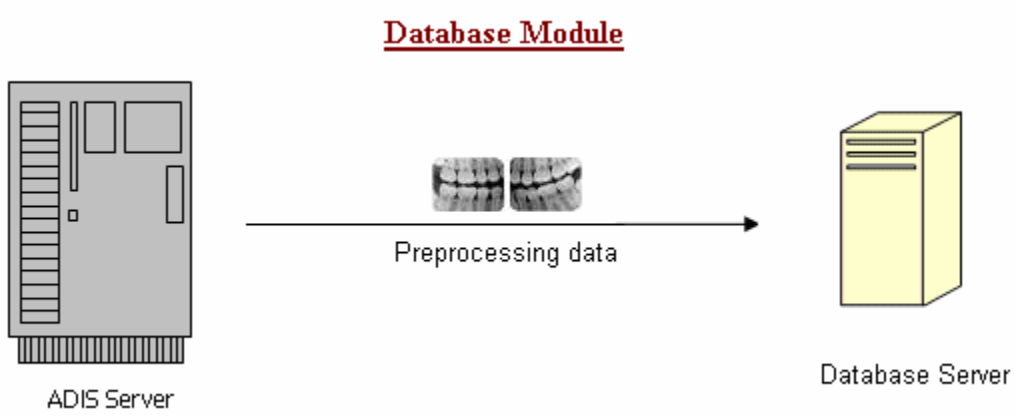


Figure 4.19: Block Diagram for Database Module

4.7.3 Working of Web-ADIS with Bridge Module:

Bridge Module is meant to facilitate a congenial environment for researchers in other universities working on ADIS to use the database in West Virginia University. They can store the high level features and can also retrieve high level features, dental records stored in the DIR, set of Case Ids with a mortality tag and preprocessing data for a Case Id by connecting to the database server hosted in West Virginia University.

This module plays a very important role to train Web ADIS environment to the realizations being developed at Michigan University and University of Miami and will make the deployment of these realizations easier on the server.

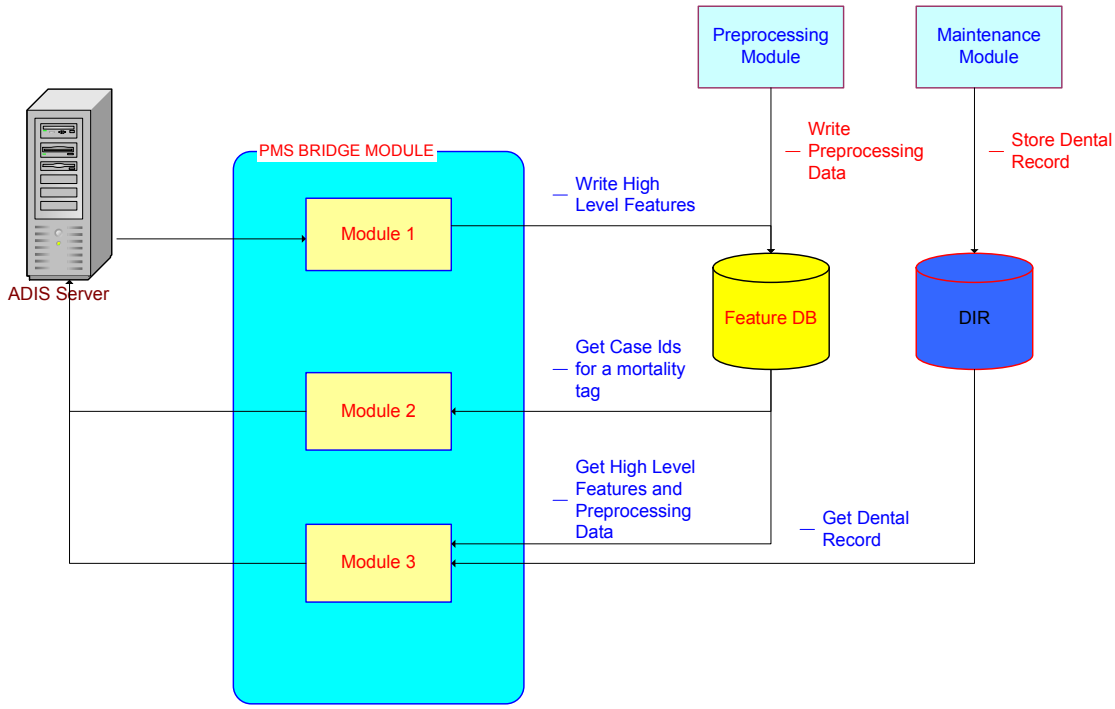


Figure 4.20: Block Diagram for bridge Module

Subcomponents of Bridge Module:

- **Module1:**

  When a researcher in other university runs his realizations on the records in the database of Web-ADIS, he gets back required parameters which are termed as High Level features. He can connect to the database server through Module1 and write these high level features on to the database so that these can be obtained back to when needed for future use.

- **Module 2:**

Module2 is meant to return an array of caseids which satisfy the constraints mentioned by the user. The constraint supported in the present version is mortality tag. The mortality tag here is AM or PM.

- **Module 3:**

  The realizations for potential match search and Image Comparison need the dental information like preprocessing data, the high level features and dental image of the reference records which are provided by Module3.

CHAPTER 5

Testing and Results

The testing for Web-ADIS was done on different test cases and scenarios. The testing of the web interface was done by verifying the working of the various components involved in different modules. The testing tool used for testing the beans used in the web application is JUnit. All components were tested and some of the problems were solved while some remained unsolved.

5.1 Testing Scenarios for Identification Mode:

5.1.1 Authentication Module:

Problems and errors encountered and methods adopted to solve:

- The authentication method used is FORM Authentication and initially the username and password were not encrypted. This is considered very insecure as the username and password can be tapped from network if not encrypted. As a remedy the form was encrypted with username and password and then decoded again at server end to ensure security.

- Rarely authentication Module gives a 'javax.servlet.ServletException: Io exception: The Network Adapter could not establish the connection' error. This is because OracleOraDb10g_home1TNSListener service is not started. OracleOraDb10g_home1TNSListener service is meant to start automatically but doesn't start at times and needs to be started manually.

Successful Test Cases:

- The user could log in into the Web-ADIS application. Only valid users who are registered in the system could log in into the system.
- Users with an invalid username or password are redirected to an error page.
- Users who login in without entering username or password are redirected to an error page which prompts to reenter the username and password.
- Users are alerted when then have a Caps Lock on when they are entering the password.

5.1.2 Identification Module:

Problems and errors encountered and methods adopted to solve:

- The identification module was allowing only search for one record for one session. The bug was in the initialization of Preprocessing Bean which was coded to get initialized only for the first time. This was changed to flexibly initialize Preprocessing Bean for multiple searches in the same session.

- This module was accepting nulls for subject records. This was solved by adding a java script which checks for subject record. It alerts the user if a record is not selected and goes on to next step only if a record is selected.
- This module was accepting files with all extensions. This was solved by adding a java script which checks for the extension of the file being uploaded. If the extension is a non-jpeg or non-jpg, it alerts the user to select a file with extension jpeg or jpg.

Successful Test Cases:

- The subject record is being uploaded onto the local disk of server from the local disk of client.
- If the client doesn't want to upload his own reference record then he is directed to page where he can see the preprocessing results of subject record and the candidate list.
- If the client wants to upload his own reference record then he is directed to page where he can upload the records.
- The options selected by the client in the Identification page is stored in the preprocessing bean which in turn is reflecting in the page where the preprocessing results of the subject record is being showed.

5.1.3 Preprocessing Module:

Problems and errors encountered and methods adopted to solve:

Matlab supports only one instance of matlab engine at a time. Server was crashing down when multiple requests for the Preprocessing module is being made simultaneously. This is because Matlab doesn't support multiple sessions running at the same time due to which multithreading couldn't be used to solve this problem. This was solved by synchronizing the request for preprocessing module.

Successful Test Cases:

- The request for Preprocessing is completed successfully for a single request and displays the results of preprocessing.
- Multiple requests for preprocessing by multiple clients are being served by the server and the results are displayed after preprocessing is done.

5.1.4 Potential Match Module:

The potential Match Module is used in two different scenarios in Web-ADIS which are as follows:

- Potential Match Module (Without uploading reference records)
- Potential Match Module (Uploading Reference Records)

Potential Match Module (Without uploading reference records) refers to the scenario where user uses reference records from the database without explicitly uploading any

reference record in the identification mode. In Potential Match Module (Uploading Reference Records) user uploads reference records to the database in the identification mode.

Potential Match Module (Without uploading reference records)

Problems and errors encountered and methods adopted to solve:

- The potential match search was giving Candidate List which contained both AM and PM records. Candidate List should contain PM records if the subject record submitted is an AM and vice-versa. This problem was solved by making some changes to the code in SQL to get the expected results only.
- The potential match search was giving erroneous results due to some bugs in the code. This was solved by after I read the literature part of the ORDImageSignature concept used and then identifying the bug. The bug was fixed to get the correct result.

Problems and errors encountered which are yet to solved:

The potential match search doesn't work for multiple requests simultaneously. Work is being done to solve this problem.

Successful Test Cases:

- The Candidate List of PMs was obtained by submitting an AM subject record.
- The Candidate List of AMs was obtained by submitting a PM subject record.

- The Candidate List of PMs with a groupid was obtained by submitting an AM subject record and a groupid from which we want the system to search in.
- The Candidate List of AMs with a groupid was obtained by submitting a PM subject record and a groupid from which we want the system to search in.

Potential Match Module (Uploading Reference Records):

Problems and errors encountered and methods adopted to solve:

- This module was accepting nulls for reference records. This was solved by adding a java script which checks for reference record. It alerts the user if a record is not selected and goes on to next step if a record is selected.
- This module was accepting in files with all extensions. This was solved by adding a java script which checks for the extension of the file being uploaded. If the extension is a non-jpeg or non-jpg, it alerts the user to select a file with extension jpeg or jpg.

Problems and errors encountered which are yet to solved:

The potential match search doesn't work for multiple requests simultaneously. Work is being done to solve this problem.

Successful Test Cases:

- Reference records were uploaded successfully onto the database. The uploaded images reflected in the Candidate List.

- The reference records are preprocessed and the results are displayed.

5.1.5 Image Comparison Module:

Problems and errors encountered and methods adopted to solve:

- Matlab supports only one instance of matlab engine at a time. Server was crashing down when multiple requests for the Image Comparison module are being made simultaneously. This is because Matlab doesn't support multiple sessions running at the same time due to which multithreading couldn't be used to solve this problem. This was solved by synchronizing the request for Image Comparison module.
- Image Comparison module is using the bridge module to get the images from database. After few iterations of Image Comparison its gives out a java.lang.OutOfMemory error. This was solved by identifying the variables which are out of out of reach for Garbage Collector and deleting them in the code.

Successful Test Cases:

- The image comparison module is invoked after the candidate list is got.
- The image comparison module computes the results which are displayed for the users to view.

5.2 Testing Scenarios for Maintenance Mode:

5.2.1 Authentication Module:

(Same as Authentication Module in section 6.1)

5.2.2 Maintenance Module:

Problems and errors encountered and methods adopted to solve:

- This module was accepting nulls for subject records. This was solved by adding a java script which checks for subject record. It alerts the user if a record is not selected and goes on to next step only if a record is selected.
- This module was accepting files with all extensions. This was solved by adding a java script which checks for the extension of the file being uploaded. If the extension is a non-jpeg or non-jpg, it alerts the user to select a file with extension jpeg or jpg.

Successful Test Cases:

- The non-dental and dental information of the reference record to be uploaded is entered and then submitted successfully. This information is stored in Maintenance bean so that these can be used for next steps.

5.2.3 Preprocessing Module:

Problems and errors encountered and methods adopted to solve:

(Same as in Section 6.1.3)

Successful Test Cases:

- The request for Preprocessing of reference record is being successfully completed for a single request and displays the results of the preprocessing.
- Multiple requests for preprocessing of reference records by multiple clients are being served by the server and the results are displayed after the preprocessing is done.

5.2.4 Database Module:

Problems and errors encountered and methods adopted to solve:

The attributes of the database segment_id and film_id were being generated different from expected. This problem was solved by fixing the bug in the code.

Successful Test Cases:

- The preprocessing information was read from preprocessing text file generated after the preprocessing of the reference record. This information was then updated on to the database.

5.3 Testing Scenarios for Bridge Module:

Problems and errors encountered and methods adopted to solve:

Module3 reads the preprocessing text file and stores all the parameters in string arrays. There was problem converting these java string arrays to arrays used in Matlab. This problem was solved by using a Matlab program which reads the preprocessing text instead of java program.

Successful Test Cases:

- Module1 writes the High Level Features, caseid and the label onto the database.
- Module2 returns all the caseids in the database which satisfies the features specified. In this case only Mortality Status is used.
- Module3 returns the image of the record, preprocessing data and the high level features for the reference record whose caseid is provided. This info is provided only if input arguments 'R','P' and 'F' are '1','1' and '1' respectively. These are not returned if '0' is provided as input for these arguments.
- Module3 returns the image of the record and the preprocessing data of the subject record whose caseid is provided. This info is provided only if input argument 'R', 'P' are '1', '1'. These are not returned if '0' is provided as input for these arguments.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions:

Web-ADIS is a web interface for using the realizations developed for Automated Dental Identification in West Virginia University, Michigan State University and University of Miami on internet. This would give the FBI agents and Forensic experts the ability to use the Automated Dental Identification System from their office desks.

This environment consists of the Identification Module, Maintenance Module and Bridge Module. Identification mode allows the users to identify the subject dental record from a set of reference dental records already in the DIR. Maintenance mode allows them to upload the reference records of their interest onto the database. Bridge module is provided to allow the researchers from Michigan State University and University of Miami to use the database in West Virginia University to extract the interested reference dental records and write the high level features on to the database. These high level features can also be extracted from the database using the Bridge Module.

It is a cross-platform environment capable of running in Windows NT or any Unix

server. It works on web browsers such as Internet Explorer, Mozilla. The integrated set of web based Java Server Pages are built on industry-standard server architectures, which allow the environment to adapt to new platforms.

6.2 Future Work:

The present version of Web-ADIS can be improved to provide other services. Future work is also needed to improve the modules developed and described in this thesis.

- The server on which the present step up of Web-ADIS is running has Preprocessing, Potential Match, Image Comparison and Database server working on the same server. The assumption made in the setup is that the server can sustain the load for all the components running on the same machine. In future as the number of records in the reference database increases the load on the server too increases, so the option would be to setup different servers for different components. When this is done not only the load gets shared between different servers but also the speed of the system increases drastically.

- The present setup invokes the matlab engine for preprocessing, potential match and Image comparison. This restricts the processing of more than two images simultaneously as more than one instance of matlab cannot be run simultaneously. In future this problem can be solved by setting up a server for the components where matlab is needed and the server should be programmed in such a way that

they run simultaneously.

- The present realizations of Preprocessing, Potential Match Search and Image Comparison Module take more time, so the user is made to wait for the long time. This can be cut down by adding a ticketing service which e-mails the users when the results are ready to be viewed. This should e-mail a valid link which would allow the users to look at the results when clicked.

- The present version of Web-ADIS has provisions for testing and System Configuration which are meant to be added in future.

- The results of preprocessing and image comparison are written on text files which are read by the java beans. The performance and Accuracy can be improved by using XML documents instead of text files.

References

[Bean-Spec] Java Beans API Specification from Sun Microsystems.

[Diaa] Diaa Eldin M. Nassar "A Prototype Automatic Dental Identification System (ADIS)" Masters Thesis, Electrical Engineering – WVU, 2001..

[En] Definition of Webserver (http://en.wikipedia.org/wiki/WebServer).

[Eyad] Eyad Haj Said, Diaa Eldin M. Nassar, Gamal Fahmy, Hany H. Ammar "Teeth Segmentation in Digitized Dental X-Ray Films using Mathematical Morphology" Department of Electrical and Computer Engineering – WVU.

[IBIMA] Satyasrinivas Chekuri, Diaa Eldin M. Nassar, Ayman A. Abaza, Ali Bahu, and Hany H. Ammar "A web-based Automated Dental Identification System (webADIS)" – IBIMA, 2005.

[Ikkry04] Mohamed Ikkery "Requirements and Design Model for ADIS completed in UML" Masters Problem Report. Department of Electrical and Computer Engineering – WVU, 2004

[J2ee] Art Taylor "J2EE and beyond – Design, Develop, and Deploy World-Class Java Software".

[Jain99] K. Jain, Rudd Bolle, Sharath Pankanti, "Biometrics: Personal Identification in Networked Society", Kluwer Academic Publishers, (15 January 1999).

[Java] Overview of the JNI (http://java.sun.com/docs/books/tutorial/native1.1/concepts/index.html)

[Java2] JavaServer Pages[tm] Technology - White paper (http://java.sun.com/products/jsp/whitepaper.html)

[Java3] Danny Goodman "JavaScript Bible" Vol 3, pp 8-10, 1998.

[Joseph] www.forensiconline.com website of a forensic pathologist named Joseph I Cohen.

[Jsp] Bryan Basham, Kathy Sierra and Bert Bates "Head First Servlets and Jsp" – 2004, Chapter-7, pp 281-294

[Mtech] Definition of Authentication (http://mtechit.com/concepts/authentication.html)

[Nassar03] Nassar D.E., Dr. Hany Ammar, "A Prototype Automatic Dental Identification System" (ADIS), Department of Electrical and Computer Science Engineering – WVU. March 2003) (http://www.digitalgovernment.org/dgrc/dgo2003/cdrom/DEMOS/adis.pdf)

[Net] JavaScript and Java (http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/)

[Oreilly] The Component Model (http://www.oreilly.com/catalog/javabeans/chapter/ch01.html)

[RM03] West Virginia University Research Corporation-West Virginia University (2003).

[Scwcd] Hanumant Deshmukh, Jignesh Malavia, Matthew Scarpino, "SCWCD Exam Study Kit" - 2002, Chapter-3, pp 176-182.

[Stan] Introduction to JDBC (http://www-db.stanford.edu/~ullman/fcdb/oracle/or-jdbc.html)

[Zanab04] Zainab Millwala "A dual-stage approach to dental image registration" Masters Thesis. Department of Electrical and Computer Engineering – WVU, 2004.

Appendix

Design and Implementation

Introduction

This chapter describes the design and implementation of Web-ADIS. The implementation of Web-ADIS is done in a three tier environment as discussed in chapter -2. SQL is used to implement the database and the database server used is Oracle 10g. Web-ADIS application is developed using J2EE, JNI and Java Script.

The design of the database is discussed in section 5.2. This section discusses the tables and the attributes used in the tables in detail. The design of the Web-ADIS web application is explained in section 5.3. The section discusses the authentication module, preprocessing module, potential match module, image comparison module and bridge module.

Design of Database

This section explains in detail the proposed structure for the database. It is necessary to understand from the starting that the web application connects and interacts with the

database using Java Database Connectivity which in turn uses suitable Database Drivers to connect to the Database.

The database used by web-ADIS is divided into two parts based on the contents being stored:

- Image Database
- Feature Database

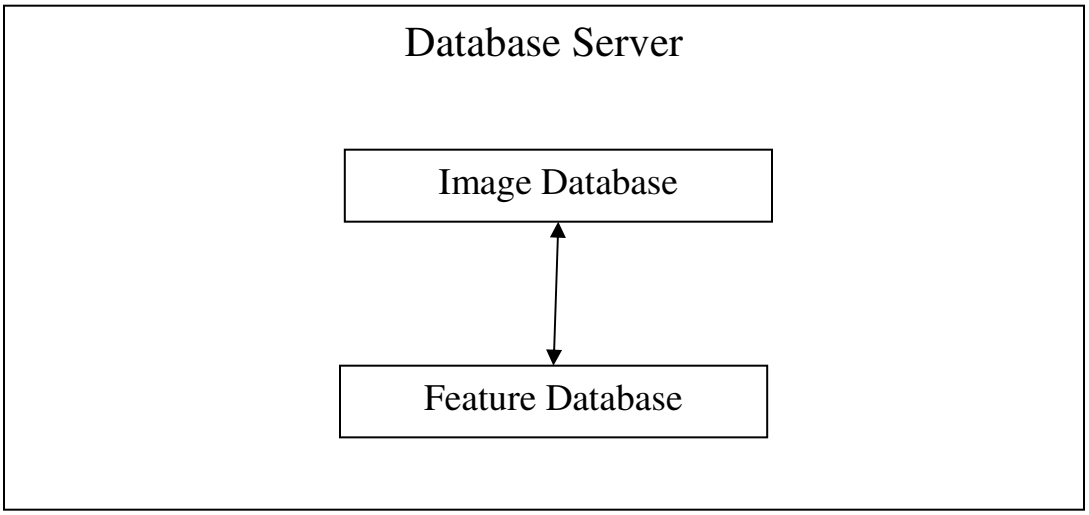


Figure 1: Block Diagram for Database Server

The Database diagram for the Image Database and the Feature Database is shown below:

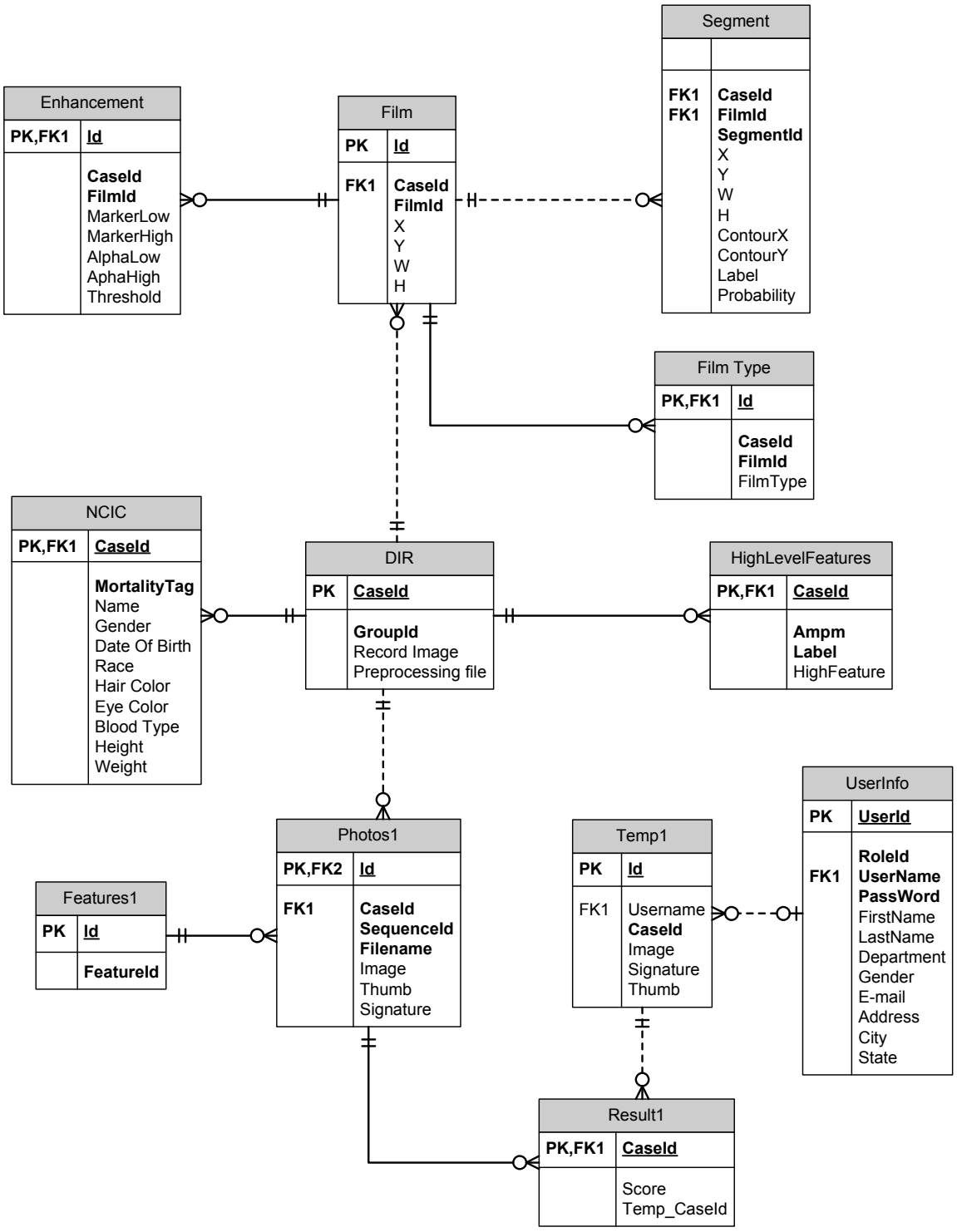


Figure 2: Design of the Database for Web-ADIS

Image Database:

This Database is used to store the all the images of the reference records. These dental images are termed as restored in rest of the chapter.

Digital Image Repository (DIR):

The dental image submitted by the user is stored in Digital Image Repository (DIR). The structure of the DIR is as shown below:

Attributes:

- **CaseId:**

  This is the primary key for DIR table. This is used to uniquely identify a dental record in the database.

- **GroupId:**

  This is used to uniquely identify the group that can be related to a mass disaster or identify the records that are uploaded by a researcher. The main aim of groupid is cut down the search time for server when search for dental records is being done on a particular interest. For example all the casualties in flood or airplane crash can be put into group identified by groupid which will help reduce the retrieval time when they are needed in future.

- **Record Image:**

This is a field which holds the image of the dental record.

- **Preprocessing File:**

  This field holds the text file with the preprocessing information. This is created to serve the interest of Bridge Module and Image Comparison Module.

Table Schema:

Attribute Name	**Data Type, Constraint**
Case Id	Number, Primary Key
Group Id	String
Record Image	ordsys.ordimage
Preprocessing File	blob

Table 1: Attributes and its features in DIR

Photos:

This is a table stores the data about the films of reference records. The data in this table pertains to data used by the potential match module.

Attributes:

- **Id:**

This is a primary key and is used to uniquely identify each record in this table. This is generated automatically by a sequence in oracle when ever a film of a record is being uploaded onto the database.

- **Case Id:**

  This is a foreign key to the Case Id attribute in DIR table.

- **Sequence Id:**

  This is a id given to uniquely identify a film in a record.

- **Group Id:**

  This is a id given to a set of reference records which fall into a particular group.

- **Filename:**

  This attribute stores the name of the record.

- **Image:**

  This attribute stores the film of the reference record.

- **Thumb:**

  This attribute stores the thumb image of the film of the reference record.

- **Signature:**

  This attribute stores the color, texture and shape information of the film.

Table Schema:

Attribute Name	**Data Type, Constraint**
Id	Number, Primary Key

Case Id	Number, Foreign Key
Sequence Id	Number
Group Id	String
Filename	String
Image	ordsys.ordimage
Thumb	ordsys.ordimage
Signature	ordsys.ordimagesignature

Table 2: Attributes and its features in Photos

Temp:

This table stores the data about the films of subject records. The data in this table pertains to data used by the potential match module.

Attributes:

- **Id:**

  This is primary key used to uniquely identify each record in this table. This is generated automatically by a sequence in oracle when ever a film of a subject record is being uploaded onto the database.

- **Username:**

  This attribute stores the username of the user who uploaded the subject record. This also plays the role of a foreign key to the username attribute of the Userinfo

table.

- **Case Id:**

  This is a unique id given to each subject record. The same case id will be used for all the films which belong to the same subject record.

- **Image:**

  This attribute stores the film of the subject record uploaded by the user.

- **Signature:**

  This attribute stores the color, texture and shape information of the film.

- **Thumb:**

  This attribute stores the thumb image of the film of the subject record.

Table Schema:

Attribute Name	**Data Type, Constraint**
Id	Number, Primary Key
Username	String, Foreign Key
Case Id	Number
Image	ordsys.ordimage
Signature	ordsys.ordimagesignature
Thumb	ordsys.ordimage

Table 3: Attributes and its features in Temp

Feature Database:

This part of the database is used to store the dental and non-dental features of the reference records. Dental features have can be preprocessing data of reference record. Non-dental feature can be personal information of the person whose dental record is being stored. This also stores the data needed to help the authentication module to work in this web interface.

Features:

This table is used by the preprocessing module developed in WVU to store the features of the reference records.

Attributes:

- **Id:** This is the primary key to uniquely identify each record in this table. This also plays the role of a foreign key to the Id attribute of the Photos table.
- **Feature Id:** This attribute descries the mortality status of the record. This takes a value 17 for Anti Mortem and 18 for Post Mortem.

Table Schema:

Attribute Name	Data Type
Id	Number,Primary Key

Feature Id	Number

Table 4: Attributes and its features in Features

Result:

The Potential Match module needs a table to temporarily store all the match scores of the subject record to reference records before presenting them to the user.

Attributes:

- **Case Id:**

  This is the primary key and also the foreign key to the CaseId attribute of the Photos table.

- **Score:**

  This attribute stores the scores of the subject record to the reference records.

- **Temp_CaseId:**

  This attribute stores the temporary case id assigned to subject record.

Table Schema:

Attribute Name	**Data Type, Constraint**
Case Id	Number, Primary Key and Foreign Key

Score	Number
Temp_CaseId	Number

Table 5: Attributes and its features in Result

UserInfo:

This table stores the information of the users who are authenticated to use the web interface for ADIS. As the authentication used by Web-ADIS is Username/ Password all the related information will be stored in this table.

Attributes:

- **UserId:**

  This is id given to every user authenticated to use the web interface and also the primary key of the table.

- **RoleId:**

  This is an id given to the set of privileges the user has on Web-ADIS interface and the database. A normal user will be assigned a roleid which has only the privilege to use the Identification Module. If the user is a forensic expert then he would have the privilege to use Identification Module as well as Maintenance Module to update the algorithms.

- **UserName:**

  This attribute stores the username of the user. This is the foreign key to

UserName attribute of Temp table.

- **PassWord:**

  This attribute stores the password chosen by the user.

- **FirstName:**

  This attribute stores the first name of the user.

- **LastName:**

  This attribute stores the last name of the user.

- **Department:**

  This attribute stores the department in which the user works.

- **Gender:**

  This attribute stores the gender of the user.

- **E-mail:**

  This attribute stores the e-mail address of the user.

- **Address:**

  This attribute stores the street address of the user.

- **City:**

  This attribute stores the city of the user.

- **State:**

  This attribute stores the state of the user.

Table Schema:

Attribute Name	Data Type
UserId	Number, Primary Key
RoleId	Number
UserName	String, Foreign Key
PassWord	String
FirstName	String
LastName	String
Department	String
Gender	String
E-mail	String
Address	String
City	String
State	String

Table 6: Attributes and its features in UserInfo

NCIC:

This table contains all the non-dental data of the owner of the dental record being uploaded into the system. Apart from helping to record the general information of the owner of the dental record uploaded, this will also help to filter the reference records using the known information of the subject record. For example if the race is known to be Asian and blood type to be A-, then the potential match search can be done only after

filtering down all the reference records whose race is not Asian and blood group is not A-

.

Attributes:

- **Case Id:**

  This is the Primary key of NCIC table and also a Foreign key to connect to DIR table.

- **Mortality Tag:**

  This is a field that describes the mortality status of the Dental record. This takes either ‘AM’ or ‘PM’ as its values.

- **Name:**

  This stores the name of the person whose dental record is being uploaded.

- **Gender:**

  This column stores the gender of the subject. This takes a value among Male or Female.

- **Date of Birth:**

  This column stores the date of birth of the person.

- **Race:**

  This attribute stores about the race of the person. The database takes an value among Negroid, Asian, Hispanic, Native American, White, Aborigine or other.

- **Hair Color:**

  This attribute stores the color of the hair of the subject. This takes one value among Bald, Black, Blond, Brown, Gray, Red and White.

- **Eye Color:**

  This attribute describes the color of eyes. This takes a value among Black, Blue, Brown, Green, Hazel, Violet and White.

- **Blood Type:**

  This the blood group of the subject. This takes a value among A-, A+, B-, B+, O-, O+, AB-, AB+.

- **Height:**

  This stores the approximate height of the person.

- **Weight:**

  This stores the approximate weight of the person.

Table Schema:

Attribute Name	**Data Type, Constraint and Description**
Case Id	Number. Primary Key and Foreign Key. Case Id is primary key to NCIC table and foreign key to DIR.
Mortality Tag	String
Name	String
Gender	String

Date of Birth	Date
Race	String
Hair Color	String
Eye Color	String
Blood Type	String
Height	String
Weight	String

Table 7: Attributes and its features in NCIC

Film:

This table stores all the preprocessing data of each film extracted from the dental reference record. The preprocessing data is got after the preprocessing steps like cropping, enhancement, segmentation, film classification, contour extraction and labeling are executed on the reference record uploaded.

Attributes:

- **Id:**

  This is the primary key to uniquely identify each record this table uniquely. Each time preprocessing data for a film in the record is to be stored in this table, the value for this column is generated automatically using a sequence.

- **Case Id:**

Each reference record uploaded on to the system is uniquely identified by a Case Id. This column also acts as a foreign key to the primary key in DIR.

- **Film Id:**

  Each film is given an id to uniquely identify it in a dental record. Each film in the database can be uniquely identified by a (Case Id, Film Id) pair.

- **X:**

  The X coordinate of each film is stored in this attribute.

- **Y:**

  The Y coordinate of each film is stored in this attribute.

- **W:**

  The width of each film is stored in this attribute.

- **H:**

  The height of each film is stored in this attribute.

Table Schema:

Attribute Name	**Data Type, Constraints and Description**
Id	Number, Primary Key
Case Id	Number, Foreign Key

Film Id	Number
X	Number. X refers to X Coordinate of the Film
Y	Number. Y refers to Y Coordinate of the Film
W	Number. W refers to Width of the Film
H	Number. H refers to Height of the Film

Table 8: Attributes and its features in Film

Film-Type:

This Table stores the Film classification information.

Attributes:

- **Id:**

  This is the primary key of this table. This also acts as a foreign key to the primary key of Film table.

- **Case Id:**

This is id given to each reference record to uniquely identify a dental record.

- **FilmId:**

  (Refer to FilmId attribute in Film Table)

- **Film Type:**

  This attribute stores the type of film that the FilmId is representing. This can take a value among bite wing, periapical view and panoramic view.

Table Schema:

Attribute Name	**Data Type, Constraints**
Id	Number. Primary Key and Foreign Key.
CaseId	Number
FilmId	Number
Film Type	String

Table 9: Attributes and its features in FilmType

Enhancement:

This table stores the enhancement information of films.

Attributes:

- **Id:**

  This acts as a primary key for this table and also plays the role of the foreign key to the primary key of the Film table.

- **CaseId:**

  This is case id of the reference record to which the film belongs.

- **Film Id:**

  (Refer to FilmId in Film Table)

- **MarkerLow:**

  This attribute stores the parameter named 'Low Marker' returned by the Enhancement module for that film in preprocessing.

- **MarkerHigh:**

  This attribute stores the parameter named 'High Marker' returned by the Enhancement module for that film in preprocessing.

- **AlphaLow:**

  This attribute stores the parameter named 'Low Alpha' returned by the Enhancement module for that film in preprocessing.

- **AlphaHigh:**

  This attribute stores the parameter named 'High Alpha' returned by the Enhancement module for that film in preprocessing.

- **Threshold:**

  This attribute stores the parameter named 'Threshold' returned by the

Enhancement module for that film in preprocessing.

Table Schema:

Attribute Name	**Data Type, Constraints and Description**
Id	Number. Primary Key, Foreign Key
Case Id	Number
Film Id	Number
MarkerLow	String. Represents 'Low Marker' Parameter
MarkerHigh	String. Represents 'High Marker' Parameter
AlphaLow	String. Represents 'Low Alpha' Parameter
AlphaHigh	String. Represents 'High Alpha' Parameter
Threshold	String. Represents 'Threshold' Parameter

Table 10: Attributes and its features in Enhancement

Segment:

This table stores all the data about all the segments of films in this table.

Attributes:

- **Case Id:**

  This is id given to each reference record to uniquely identify a dental record. This acts as a foreign key to the Case Id attribute of the Film table.

- **Film Id:**

  Each film is given an id to uniquely identify it in a dental record. This acts as a foreign key to the Film Id attribute of the Film table.

- **Segment Id:**

  This is a id given to uniquely identify the segment in a film. Each segment can be uniquely identified by (Case Id, Film Id, Segment Id).

- **X:**

  This attribute represents the X-coordinate of the segment in the film.

- **Y:**

  This attribute represents the Y-coordinate of the segment in the film.

- **W:**

  This attribute represents the width of the segment in the film.

- **H:**

  This attribute represents the height of the segment in the film.

- **ContourX:**

  This attribute represents an array of X-coordinates of the contour of segments in the film.

- **ContourY:**

  This attribute represents an array of Y-coordinates of the contour of segments in the film.

- **Label:**

  This attribute represents the label of the segment.

- **Probability:**

  This attribute represents the probability.

Table Schema:

Attribute Name	**Data Type**
Case Id	Number, Foreign Key
Film Id	Number, Foreign Key
Segment Id	Number
X	String
Y	String
W	String

H	String
ContourX	String[]
ContourY	String[]
Label	String
Probability	String

Table 11: Attributes and its features in Segment

High Level Features:

This table is meant for storing all the high level features. This is meant to be used by the users from other universities through the bridge module provided by the West Virginia Web-ADIS Research team.

Attributes:

- **Case Id:** This is a Primary key to uniquely identify a record in High Level Features Table. This also plays a role of foreign key to the Case Id attribute in DIR table.
- **Ampm:** This attribute describes the mortality status. This takes a value among AM (Anti Mortem) and PM (Post Mortem).
- **Label:** This attribute describes the university that uploaded the high level features for that CaseId. This takes a value among MSU (Michigan State University) and UM (University of Miami).

- **HighFeature:** This is an attribute which allows the high level features to be uploaded onto the database as an array of strings.

Table Schema:

Attribute Name	**Data Type**
Case Id	Number. Primary Key and Foreign Key
Ampm	String
Label	String
HighFeature	String[]

Table 12: Attributes and its features in HighLevelFeatures

Design of Web-ADIS

This section describes the design of different modules in the Web-ADIS. The modules different modules are Authentication Module, Preprocessing Module, Potential Match Module and Image Comparison module.

Design of Authentication Module:

This module is responsible for verifying the authentication of the users who try to login

into the secured Web- ADIS system. The users are prompted to login using a Username/Password. The Authentication will compare the Username and Password combination with the combinations in the database. If the user enters a valid combination he/she will be allowed to access the contents of the web application, otherwise the user is prompted to reenter the combination.

The Class Diagram for the Authentication Module is as given below:

Figure 3: Class Diagram for Authentication Module

Description of the Class Diagrams:

When the user first sends a request to the Web Server he will be seeing the index_jsp. When he enters a Username/Password the data entered will be sent to Login_jsp. Login_jsp with the help of SatyaBean1 compares the Username/Password with the combinations. If either of the fields or both are left blank Login_jsp redirects the control to reentry_null_jsp. If the Username/Password combination entered by the user is wrong then Login_jsp redirects to reentry_error_jsp page. If the Username/Password combination is correct Login_jsp redirects to index1_jsp. index1_jsp gives access the other modules by using home_jsp, systemConfiguration_jsp, Identification_jsp and Maintenance_jsp pages.

Class: index_jsp

This class consists of two text fields meant for Username and Password respectively. The user enters the Username and Password and clicks the Login button in the page.

Class: Login_jsp

This class takes in the username and password entered in the index_jsp. If it finds that the either of username or password or both are not entered then it redirects to reentry_null_jsp. If the username and passwords are entered then it sends the information to SatyaBean1 for verification. If the username/password combination is correct then it redirects to index1_jsp, otherwise it redirects to reentry_error_jsp. If the

username/password combination is correct then it redirects to index1_jsp which inturn gives access to home_jsp, systemConfiguration_jsp, Identification_jsp and Maintenance_jsp pages.

Class: reentry_null_jsp

This class prompts the user to enter the username and password with the message that the both these fields cannot be left empty.

Class: reentry_error_jsp

This class prompts the user to enter the username and password with the message that the previous combination of username/password was wrong.

Class: SatyaBean1

This is a bean class which is meant to process the request sent by Login_jsp to verify the username/password.

Attributes:

- **UserName**

  This attribute is used to store the username of the user who logged in.

- **JDBC_CONNECT_STRING**

  This attribute is used to store connect string with a thin driver.

- **JDBC_USER_NAME**

This is used to store the username of the database.

- **JDBC_PASSWORD**

  This is used to store the password of the database.

Methods:

- **getConnection( )**

  This is the main method for Java Database Connectivity. This method does the job of connecting the bean to database to make the interaction of web application with the database possible. The interaction here can be insertions into the tables, deletions from the tables and using the sequences.

- **identifyUser( )**

  This is the method which does the job of verifying the username and the password. This takes in two input arguments username and password which are entered by the user. It returns a '1' if the username/password combination is right and returns a '0' otherwise.

Class: index1_jsp

This is the first page which gets displayed when the user logins in successfully. This is a page with multiple frames.

Class: home_jsp

This is the page which briefs the users about Automated Dental Identification System.

Class: systemConfiguration_jsp

This is the page which allows the user to select the configuration of algorithms which is to be used by the Identification Module. This is for future and is not implemented yet.

Class: Identification_jsp

This is the page which allows the user to use the Identification Module.

Class: Maintenance_jsp

This is the page which allows the user to upload the reference records on to the database.

Design of Preprocessing Module

This module is responsible for uploading the subject record and to apply the preprocessing steps by invoking Matlab.

The Class Diagrams for Preprocessing Module is as shown below:

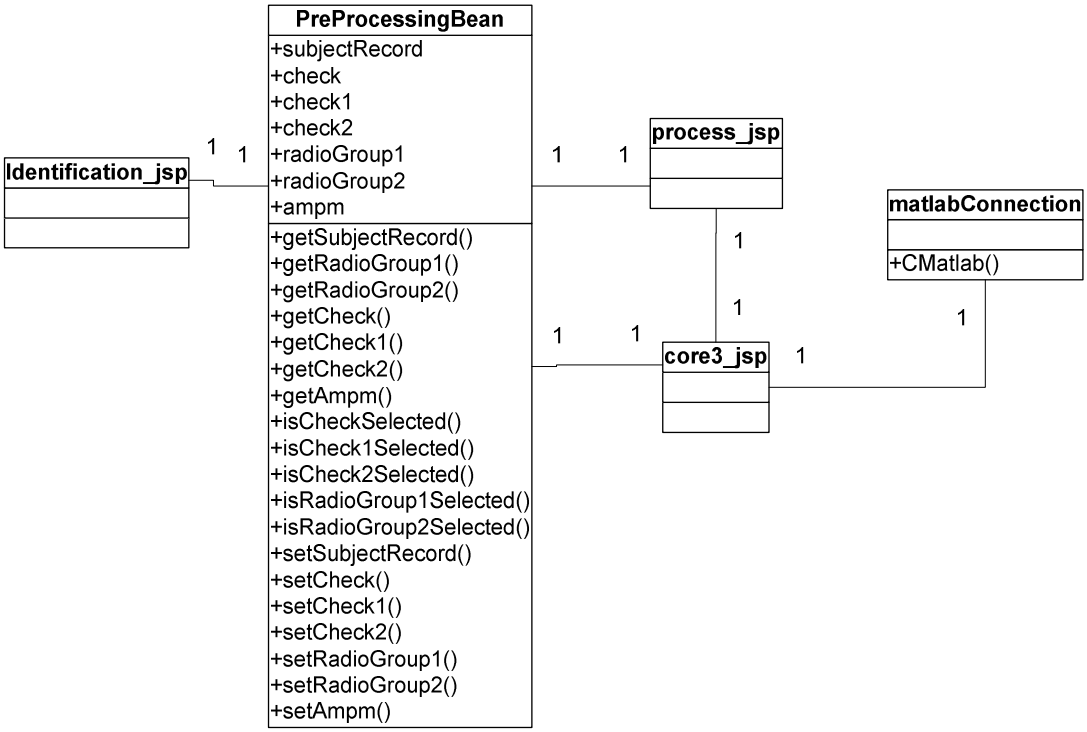


Figure 4: Class Diagram for Preprocessing Module

Description of the Class Diagram:

The user selects the options he wants in the Identification_jsp and also selects the subject record for which he needs the Match List. The options selected by the user are recorded by the PreProcessingBean to be used by other classes. After the PreprocessingBean is done with it job the process_jsp uploads the subject record from the local disk of user to the server. The command then passes over to core3_jsp which invoke Matlab by communicating with matlabConnection and PreProcessingBean.

Class: Identification_jsp

There are various options given in this page that the user can select to make use of the system according to his requirements. The options that can be selected by the user are as given below:

The mode of operation can be selected among Identification and Testing. The default value selected is Identification.

The view can be selected among End-to-End or Inspection. The default value selected is Inspection.

The subject record can be selected from the local disk of the user. The user also needs to select the mortal status of the subject record being uploaded. The user is also given the option of uploading his own set of reference records that he can upload before getting the Candidate List.

This also gives the user the option to select the different preprocessing steps that needs to be performed.

Class: PreProcessingBean

This class is meant to record the preferences selected by the user in Identification_jsp page. This will help the values to be accessed by other classes if needed for future processing.

Attributes:

- **subjectRecord**

This attribute stores the name of the subject record that is being uploaded.

- **check**

  This will store the preprocessing steps that are selected among Record Cropping, Film Classification, Film Enhancement, Teeth Segmentation, Teeth Labeling, Teeth Contour which come under Preprocessing section.

- **check1**

  This will store the selected steps from Potential Match section.

- **check2**

  This will store the options selected among ROI Alignment, ROI Normalization, ROI Compression, Teeth Labeling, Feature Extraction, Micro Decision, Macro Decision which come under Image Comparison section.

- **radioGroup1**

  This will store the radio button selected among Identification and Testing.

- **radioGroup2**

  This will store the radio button selected among End-to-End and Inspection.

- **ampm**

  This will store the mortality status of the subject record.

Methods:

- **getSubjectRecord()**

  This will return the name of the subject record.

- **getRadioGroup1()**

This will return the value stored by radioGroup1 attribute.

- **getRadioGroup2()**

  This will return the value stored by radioGroup2 attribute.

- **getCheck()**

  This will return values stored by check attribute.

- **getCheck1()**

  This will return values stored by check1 attribute.

- **getCheck2()**

  This will return values stored by check2 attribute.

- **getAmpm()**

  This will return value stored by ampm attribute.

- **isCheckSelected()**

  This will check if at least one value is stored in check attribute.

- **isCheck1Selected()**

  This will check if at least one value is stored in check1 attribute.

- **isCheck2Selected()**

  This will check if at least one value is stored in check2 attribute.

- **isRadioGroup1Selected()**

  This will check if at least one value is stored in radioGroup1 attribute.

- **isRadioGroup2Selected()**

  This will check if at least one value is stored in radioGroup2 attribute.

- **setSubjectRecord()**

This will assign the name of the subject record which is selected by the user to the subjectRecord attribute.

- **setCheck()**

  This will assign the selected values to check attribute.

- **setCheck1()**

  This will assign the selected values to check1 attribute.

- **setCheck2()**

  This will assign the selected values to check2 attribute.

- **setRadioGroup1()**

  This will assign the selected value to radioGroup1 attribute.

- **setRadioGroup2()**

  This will assign the selected value to radioGroup2 attribute.

- **setAmpm()**

  This will assign the mortalityTag value to ampm attribute.

Class: process_jsp

This class does the job of uploading the dental record on to the server. The dental record is physically needed on the local hard disk to do the preprocessing steps on the record. This also examines whether user wants to upload his choice of reference records and accordingly passe the command to other files.

Class: matlabConnection

This class does the job of invoking the matlab engine using Java Native Interface.

Methods:

CMatlab( )

This method calls the preprocessing method from the matlab which will run the preprocessing steps on the dental record.

Class: core3_jsp

This class gets the subject record by communicating with PreProcessingBean and then invokes Matlab to run preprocessing steps.

Design of Potential Match Module

The main objective of this module is to upload the subject films got from preprocessed subject record on to the database to initiate the potential match process. After the potential match module is done matching the subject record to all the reference records in the database it gives back the candidate list. The results of Preprocessing of the subject record along with the candidate list it displayed to the user for viewing.

The Class Diagram for Potential Match Module is as shown below:

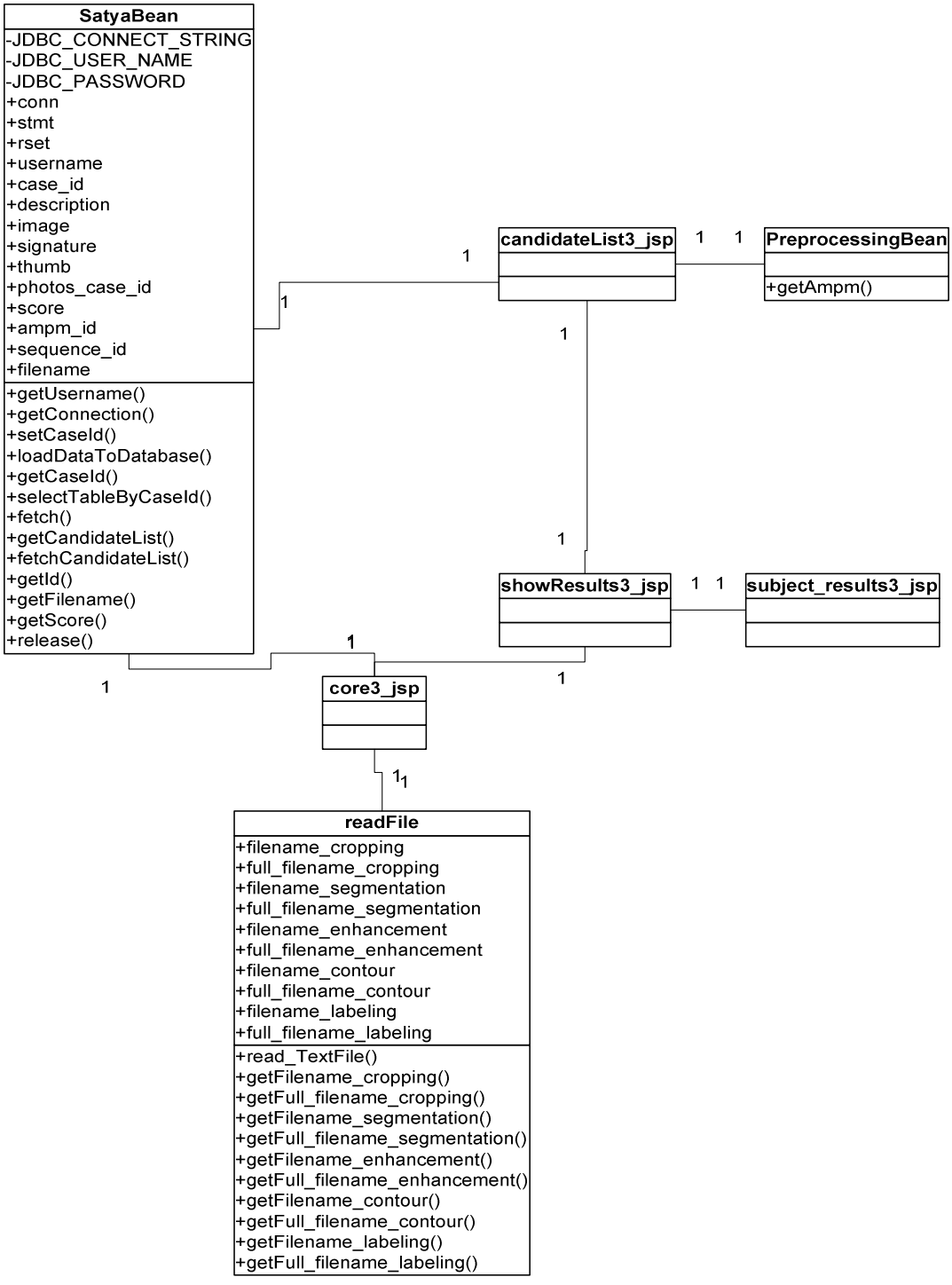

SatyaBean
-JDBC_CONNECT_STRING
-JDBC_USER_NAME
-JDBC_PASSWORD
+conn
+stmt
+rset
+username
+case_id
+description
+image
+signature
+thumb
+photos_case_id
+score
+ampm_id
+sequence_id
+filename
+getUsername()
+getConnection()
+setCaseId()
+loadDataToDatabase()
+getCaseId()
+selectTableByCaseId()
+fetch()
+getCandidateList()
+fetchCandidateList()
+getId()
+getFilename()
+getScore()
+release()
candidateList3_jsp
PreprocessingBean
+getAmpm()
showResults3_jsp
subject_results3_jsp
core3_jsp
readFile
+filename_cropping
+full_filename_cropping
+filename_segmentation
+full_filename_segmentation
+filename_enhancement
+full_filename_enhancement
+filename_contour
+full_filename_contour
+filename_labeling
+full_filename_labeling
+read_TextFile()
+getFilename_cropping()
+getFull_filename_cropping()
+getFilename_segmentation()
+getFull_filename_segmentation()
+getFilename_enhancement()
+getFull_filename_enhancement()
+getFilename_contour()
+getFull_filename_contour()
+getFilename_labeling()
+getFull_filename_labeling()

Figure 5: Class Diagram for Potential Match Module

Description of Class Diagram:

core3_jsp does the necessary steps to set the needed environment for potential match core3_jsp communicates with SatyaBean to these tasks. showResults3_jsp comprises of two pages subject_results3_jsp and candidateList3_jsp. Subject_results3_jsp is responsible to display the preprocessing results of all the subject record. candidateList3_jsp communicates with SatyaBean and PreprocessingBean to get the candidate list from the database and displays the candidate list for the user.

Class: core3_jsp

core3_jsp generates a temporary case id for the new subject record. It uploads the enhanced images into the database to set the stage for the potential match search.

Class: SatyaBean

This class offers the services to do the tasks to get the database ready for Potential Match and after the Database Server is done with the Potential Match search, its services gets back the candidate list along with the score with respect to subject record.

Attributes:

- **JDBC_CONNECT_STRING:**

  This is a connect string used to connect to Database server using a thin driver.

- **JDBC_USER_NAME:**

  This attribute stores the username of the database where the reference records are stored.

- **JDBC_PASSWORD:**

  This attribute stores the password of the database where the reference records are stored.

- **conn:**

  This attribute is of type OracleConnection and is used to get the connection to Oracle database using Java Database Connectivity.

- **stmt:**

  This attribute is of type OraclePreparedStatement and is used to execute a SQL statement after getting connected to Oracle Database.

- **rset:**

  This attribute of type OracleResultSet that is used to store all the results returned by the oracle database after the OraclePreparedStatement is executed.

- **username:**

  This attribute is used to store the username of the user who logged into the Web-ADIS web application.

- **Image:**

  This attribute is of type OrdImage and is used to temporarily store any images before they are uploaded onto the database or extracted from database.

- **Signature:**

This attribute of type OrdImageSignature is used to generate a signature of the image based on color, location, shape and texture of the image.

- **Thumb:**

  This attribute is of type OrdImage and is used to store the thumb image of the films.

- **Photos_case_id:**

  This attribute stores the Case Ids of the records in the photos table.

- **Score:**

  This attribute stores the scores of the reference record when they are compared with the subject record.

- **Ampm_id:**

  This attribute stores the id used for AMs and PMs. This takes a value of 17 for AMs and 18 for PMs. This notation is used for the tables that take part in Potential Match Search.

- **Sequence_id:**

  This is id given to the each film in record.

- **Filename:**

  This attribute stores the filename of the record.

Methods:

- **getConnection():**

  This method is used connect to the Database Server. This method plays a very

important role in Database Connectivity.

- **getNewCaseId():**

  This method is used to generate a new Case Id each time a subject record is uploaded in Identification Module and reference record is uploaded in Maintenance Module.

- **getUsername( ):**

  This method is used to get the username of the user who logged into the Web-ADIS system.

- **setCaseId():**

  This method is used to set the case_id attribute with the new Case Id generated for the subject record uploaded.

- **loadDataToDatabase():**

  This method is used to upload the cropped and enhanced films of the subject record into the database.

- **getCaseId():**

  This is used to get the value of the case_id attribute.

- **selectTableByCaseId():**

  This method is used to get the details of the subject record uploaded in that session.

- **fetch():**

  This method is used to store the data of the subject record into the local variables in SatyaBean.

- **getCandidateList():**

  This method calls the ‘adis’ procedure in Oracle to start the Potential Match Search.

- **fetchCandidateList():**

  This method fetches the Candidate List after the Database Server is done with the Potential Match Server.

- **getFilename():**

  This method returns the filename of the reference record.

- **getScore():**

  This method returns the score got when subject record is compared with a reference record.

- **release():**

  This method clears the buffer used by rset and stmt attributes. This also releases the Database connection after the Potential Match Search is finished.

Class: readFile

Preprocessing Module writes the preprocessing data of the dental record onto a file. This class reads the text file and sorts the data into cropping, enhancement, labeling, segmentation and contour.

Attributes:

- **Filename_cropping:**

This attribute stores the filenames of all the cropped films. Ideally the number of cropped images is equal to the number of films in the dental record.

- **Full_filename_cropping:**

  This attribute stores the pathname of cropped images that are stored on the local disk of server.

- **Filename_segmentation:**

  This attribute stores the filenames of all the segmented films. Ideally the number of segmented images is equal to the number of films in the dental record.

- **Full_filename_segmentation:**

  This attribute stores the pathname of segmented images that are stored on the local disk of server.

- **Filename_enhancement:**

  This attribute stores the filenames of all the enhanced films. Ideally the number of enhanced images is equal to the number of films in the dental record.

- **Full_filename_ enhancement:**

  This attribute stores the pathname of enhanced images that are stored on the local disk of server.

- **Filename_contour:**

  This attribute stores the filenames of all the films with their contours. Ideally the number of images with their contours is equal to the number of films in the dental record.

- **Full_filename_contour:**

This attribute stores the pathname of images with their contours that are stored on the local disk of server.

- **Filename_labeling:**

  This attribute stores the filenames of all the labeled films. Ideally the number of labeled images is equal to the number of films in the dental record.

- **Full_filename_ labeling:**

  This attribute stores the pathname of labeled images that are stored on the local disk of server.

Methods:

- **Read_TextFile():**

  This method sorts the data from the preprocessing file that is written by the Preprocessing Module.

- **getFilename_cropping():**

  This method returns the value of filename_cropping attribute.

- **getFull_filename_cropping():**

  This method returns the value of full_filename_cropping attribute.

- **getFilename_segmentation():**

  This method returns the value of filename_segmentation attribute.

- **getFull_filename_segmentation():**

  This method returns the value of full_filename_segmentation attribute.

- **getFilename_enhancement():**

This method returns the value of filename_enhancement attribute.

- **getFull_filename_enhancement():**

  This method returns the value of full_filename_enhancement attribute.

- **getFilename_contour():**

  This method returns the value of filename_contour attribute.

- **getFull_filename_contour():**

  This method returns the value of full_filename_contour attribute.

- **getFilename_labeling():**

  This method returns the value of filename_labeling attribute.

- **getFull_filename_labeling():**

  This method returns the value of full_filename_labeling attribute.

Class: showResults3_jsp

This shows the preprocessing results of the subject record and the Candidate List. The preprocessing results are shown with the help of subject_results3_jsp while the Candidate List is shown by candidateList3_jsp.

Class: subject_results3_jsp

This shows the preprocessed results of the subject record submitted by the user. The results shown here depend on the steps selected to be shown by the user in Preprocessing Module. By default all preprocessing steps selected will be displayed. The preprocessing steps here can be cropping, enhancement, segmentation, labeling and contours. When the

user clicks on any of the preprocessing steps, the filenames of the image files which are results of the preprocessing step selected gets displayed. The image gets displayed if anyone of the filenames is clicked.

Class: candidateList3_jsp

This shows the list of all the records with the filenames and their corresponding scores with respect to subject record. The list is sorted in ascending order of the scores. This says that the record at the top of the list is the closest match to the subject record and the difference of reference record to subject record widens as we go down the list.

Class: PreprocessingBean:

This class stores the information given by the user in the Preprocessing Module.

Methods:

- **getAmpm():**

  This returns the mortality tag of the subject record.

Design of Potential Match Module (User Uploading Reference Records)

Apart from allowing the upload of subject record, this module also allows the uploading of reference records. After the user is done uploading the reference records, he can request start of Potential Match Search. Later, the preprocessing results of the subject

record, reference record uploaded and the Candidate List gets displayed.

The Class Diagram for the Preprocessing Module used with after uploading of reference records is as shown below:

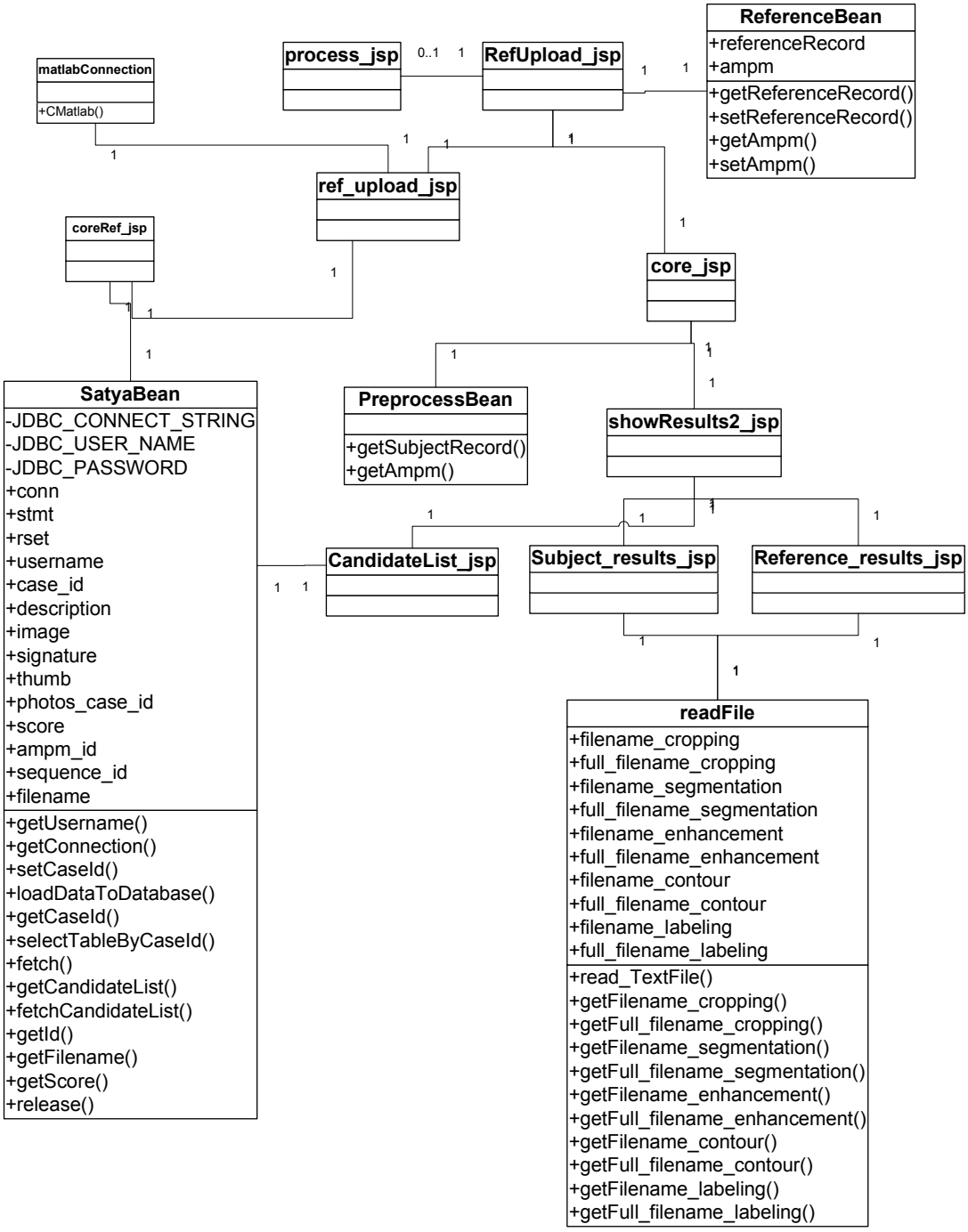


Figure 6: Class Diagram for Potential Match (With Reference Records Upload)

Description of Class Diagram:

Process_jsp redirects the command to RefUpload_jsp if the user wants to upload some reference records before doing a Potential Search on them. RefUpload_jsp, ReferenceBean, ref_upload_jsp, coreRef_jsp helps to do the upload the reference record on to the server and later uploads the enhanced films onto the database after preprocessing the dental record. Preprocessing is done with the help of matlabConnection class. After the reference records are uploaded, potential match Search is initiated with the help of SatyaBean and later the results are shown to the user by CandidateList_jsp, Subject_results_jsp and Reference_results_jsp.

Class: process_jsp

This class assess whether the user wants to upload the reference records before Potential Match Search.

Class: RefUpload_jsp

This class takes in the details about the reference records the user wants to upload.

Class: ReferenceBean

This class stores the details about the reference records that the user has uploaded. These details are later used to update the database.

Attributes:

- **referenceRecord:**

  This attribute stores the name of the reference record uploaded.

- **ampm:**

  This attribute stores the mortality status of the reference record that is uploaded.

Methods:

- **getReferenceRecord():**

  This attribute returns the value of the attribute referenceRecord.

- **setReferenceRecord():**

  This attribute sets the value of the attribute referenceRecord.

- **getAmpm():**

  This attribute returns the value of the attribute ampm.

- **setAmpm():**

  This attribute sets the value of the attribute ampm.

Class: ref_upload_jsp

This class uploads the reference record physically from the local disk of user to the local hard disk of server.

Class: coreRef_jsp

This class communicates with ReferenceBean, matlabConnection, SatyaBean and readFile classes to run the preprocessing algorithms on the reference records and to

upload the enhanced films onto the database.

Class: matlabConnection

(Refer to Class: matlabConnection in section 5.3.2)

Class: SatyaBean

(Refer to Class: SatyaBean in section 5.3.3)

Class: readFile

(Refer to Class: readFile in section 5.3.3)

Class: showResults2_jsp

This shows preprocessing results of the subject record submitted, preprocessing results of the reference records submitted and the Candidate List. The preprocessing results are shown with the help of Subject_results_jsp and Reference_results_jsp while the Candidate List is shown by CandidateList1_jsp.

Class: PreprocessingBean

This class stores the information given by the user in the Preprocessing Module.

Methods:

- **getAmpm():**

This returns the mortality tag of the subject record.

- **getSubjectRecord():**

  This returns the name of the subject record uploaded by the user.

Class: Subject_results_jsp

(Refer to Class: subject_results3_jsp in section 5.3.3)

Class: Reference_results_jsp

This shows the preprocessed results of the reference record submitted by the user. The results shown here depend on the steps selected to be shown by the user in Preprocessing Module. By default all the preprocessing steps selected will be displayed. The preprocessing steps here can be cropping, enhancement, segmentation, labeling and contours. When the user clicks on any of the preprocessing steps, the filenames of the image files which are results of the preprocessing step selected gets displayed. The image gets displayed if anyone of the filenames is clicked.

Class: CandidateList1_jsp

(Refer to Class: candidateList3_jsp in section 5.3.3)

Design of Image Comparison Module:

This module uses the Case Ids in the Candidate List as input to Image Comparison

module. This gets back the results after Image Comparison module is done with its work.

The class diagram for Image Comparison is as shown below:

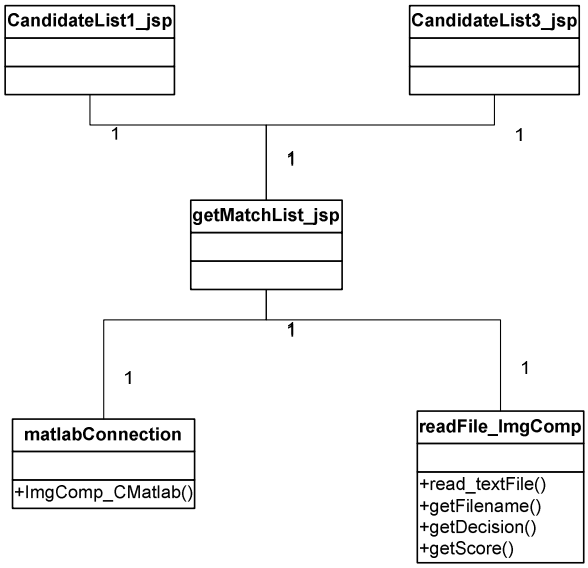


Figure 7: Class Diagram for Image Comparison Module

Description of Class Diagram:

CandidateList1_jsp displays the candidate list in potential match module with reference records being uploaded by the user, while CandidateList3_jsp displays the candidate list in potential match module where reference records are not uploaded by users. The candidate list given by CandidateList1_jsp and CandidateList3_jsp are given to Image Comparsion module by getMatchList_jsp. Image Comparison module is invoked by matlabConnection. The match list gets displayed by getMatchList_jsp with the help of readFile_ImgComp.

Class: CandidateList1_jsp

This displays the Candidate List in potential match module where reference records are uploaded by the user before getting the candidate list. This also writes the candidate list onto a file which can be used by Image Comparison module.

Class: CandidateList3_jsp

This displays the Candidate List in potential match module where reference records are not uploaded by the user before getting the candidate list. This also writes the candidate list onto a file which can be used by Image Comparison module.

Class: matlabConnection

This class invokes Matlab using Java Native Interface (JNI). As the Image Comparison module needs Matlab for running its realizations it needs to be invoked by java.

Methods:

- **ImgComp_CMatlab()**

  This method calls the Matlab method to start the Image Comparison process.

Class: readFile_ImgComp

This class reads the results written onto a file by the Image Comparison Module. This class sorts the data in the file to get the filename, score and decision.

Attributes:

- **filename:**

  This attribute stores the filenames of all the reference records in the Match List that is returned by the Image Comparison Module.

- **score:**

  This attribute stores the score of reference record when compared to subject record by the Image Comparison Module.

- **decision:**

  This attribute stores the decision which can be one among match, not match or undecided.

Methods:

- **read_textFile():**

  This methods reads the file and sorts the data to store useful information to be retrieved later.

- **getFilename():**

  This method returns the value of filename attribute.

- **getScore():**

  This method returns the value of score attribute.

- **getDecision():**

  This method returns the value of decision attribute.

Class: getMatchList_jsp

This class calls the matlabConnection to invoke the Image Comparsion Module and then reads the results given by the module with the help of read_TextFile class. This class displays these results to the user.

Design of Maintenance Module:

Maintenance Module's main task is to help maintain DIR by uploading the reference records. When the operator uploads the dental reference record all the preprocessing steps are done on the record before uploading the record onto the database. Along with the dental record the preprocessing data also gets uploaded onto the database.

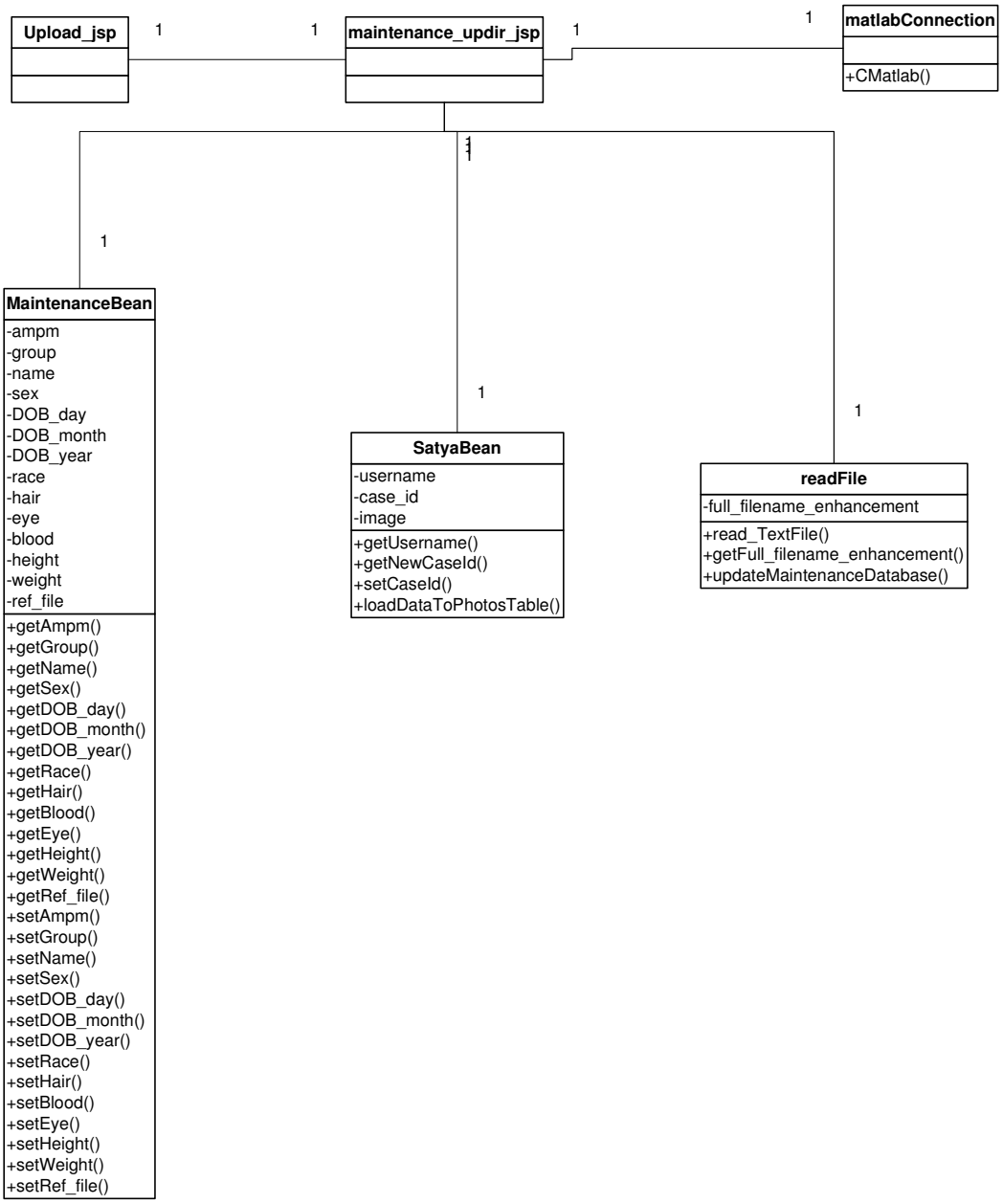


Figure 8: Class Diagram for Maintenance Module

Description of Class Diagram:

The process of uploading of reference record is initiated by Upload_jsp. Apart from

reference record this takes in non-dental details of the reference record. MaintenanceBean stores all the details entered by the user temporarily before they are uploaded onto the database. Preprocessing is done on the dental record by matlabConnection. Preprocessing data gets updated in the database by SatyaBean and readFile.

Class: Upload_jsp

This is a page which takes in needed dental and non-dental information from the user about the reference record being uploaded. The information thats requested is mortality tag, name, age, Date of Birth, Race, Hair Color, Eye Color, Blood Type, Height and Weight. This information provided depends on availability and is not onsidered mandatory for uploading the reference record.

Class: maintenance_updateDir_jsp

This class uploads the reference record from the local disk of user to the local disk of server. It runs preprocessing steps on the reference record with the help of matlabConnection class. Later SatyaBean and readFile help to update the database with the preprocessing data, reference record and the enhanced films to aid the present Potential Match realization being used by WVU.

Class: MaintenanceBean

This class keeps track of all the information given by the user in Upload_jsp.

Attributes:

- **Ampm:**

  This attribute stores the mortality tag of reference record being uploaded.

- **Group:**

  This attribute stores the group in which the reference record is being uploaded.

- **Name:**

  This attribute stores the name of the person whose dental radiograph is being uploaded.

- **Sex:**

  This attribute stores the gender of the person whose dental radiograph is being uploaded.

- **DOB_day:**

  This attribute stores the day of birth of the person whose dental radiograph is being uploaded.

- **DOB_month:**

  This attribute stores the month of birth of the person whose dental radiograph is being uploaded.

- **DOB_year:**

  This attribute stores the year of birth of the person whose dental radiograph is being uploaded.

- **Race:**

  This attribute stores the race of the person whose dental radiograph is being

uploaded.

- **Hair:**

  This attribute stores the hair color of the person whose dental radiograph is being uploaded.

- **Eye:**

  This attribute stores the eye color of the person whose dental radiograph is being uploaded.

- **Blood:**

  This attribute stores the blood group of the person whose dental radiograph is being uploaded.

- **Height:**

  This attribute stores the height of the person whose dental radiograph is being uploaded.

- **Weight:**

  This attribute stores the weight of the person whose dental radiograph is being uploaded.

- **Ref_File:**

  This attribute stores the name of the reference record that is being uploaded onto the database.

Methods:

- **getAmpm():**

This method returns the value of ampm attribute.

- **getGroup():**

  This method returns the value of group attribute.

- **getName():**

  This method returns the value of name attribute.

- **getSex():**

  This method returns the value of sex attribute.

- **getDOB_day():**

  This method returns the value of DOB_day attribute.

- **getDOB_month():**

  This method returns the value of DOB_month attribute.

- **getDOB_year():**

  This method returns the value of DOB_year attribute.

- **getRace():**

  This method returns the value of race attribute.

- **getHair():**

  This method returns the value of hair attribute.

- **getEye():**

  This method returns the value of eye attribute.

- **getBlood():**

  This method returns the value of blood attribute.

- **getHeight():**

This method returns the value of height attribute.

- **getWeight():**

  This method returns the value of weight attribute.

- **getRef_File():**

  This method returns the value of ref_File attribute.

- **setAmpm():**

  This method sets the value of ampm attribute.

- **setGroup():**

  This method sets the value of the attribute group.

- **setName():**

  This method sets the value of the attribute name.

- **setSex():**

  This method sets the value of the attribute sex.

- **setDOB_day():**

  This method sets the value of the attribute DOB_day.

- **setDOB_month():**

  This method sets the value of the attribute DOB_month.

- **setDOB_year():**

  This method sets the value of the attribute DOB_year.

- **setRace():**

  This method sets the value of the attribute race.

- **setHair():**

This method sets the value of the attribute hair.

- **setEye():**

  This method sets the value of the attribute eye.

- **setBlood():**

  This method sets the value of the attribute blood.

- **setHeight():**

  This method sets the value of the attribute height.

- **setWeight():**

  This method sets the value of the attribute weight.

- **setRef_File():**

  This method sets the value of the attribute ref_File.

Class: matlabConnection

(Refer to Class: matlabConnection in section 5.3.2)

Class: SatyaBean

This class helps to load the films of the reference records onto the database so that these can be used by the potential match realization developed in WVU.

Attributes:

- **username:**

  This attribute stores the username of the user who is logged into the system.

- **case_id:**

  This attribute stores the Case Id of the reference record.

- **image:**

  This attribute stores the films of the reference record.

Methods:

- **getUsername():**

  This method returns the value stored in the attribute username.

- **getNewCaseId():**

  This method generates a new Case Id for the reference record being uploaded.

- **setCaseId():**

  This method sets a value to case_id attribute.

- **loadDatatoPhotosTable():**

  This method uploads the films got from the reference record onto the database.

Class: readFile

This class reads the preprocessing data from the text file which is written by the preprocessing module and will upload the data onto the database.

Attributes:

- **Full_filename_enhancement:**

  This attribute stores the pathname of enhanced images that are stored on the local

disk of server.

Methods:

- **read_TextFile():**

  This method sorts the data from the preprocessing file that is written by the Preprocessing Module.

- **getFull_filename_enhancement():**

  This method returns the attribute full_filename_enhancement

- **updateMaintenanceDatabase():**

  This method uploads all the preprocessing data onto the database.

Design of Bridge Module:

This module was built to enable researchers from other universities to use the services of the database server based in West Virginia. The researchers can first get the array of Case Ids for the features they want from database by using Module2. After they got the Case Ids with the required features they can write the high level features and then read high level features, preprocessing data and get the image of the record for the Case Ids by using Module1 and Module3 respectively.

Design of Module1:

This module helps to write the high level features to the database. Each University is given a unique label to be used by their researchers while writing high level features to the database so that they can retrieve them when needed. The class diagram for Module1 is as shown below:

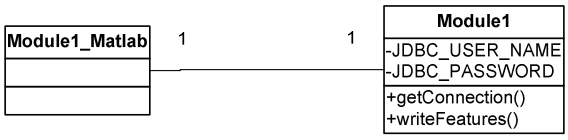


Figure 9: Class Diagram for Module1

Description of Class Diagram:

Module1_Matlab is a Matlab file which calls methods in Module1 when it wants to write the high level features to the database.

Class: Module1_matlab

This is a matlab file which makes an object out of Module1 class and uses it.

Class: Module1

This is the class which connects to the database and writes the high level features onto it.

Attributes:

- **JDBC_USER_NAME:**

This attribute stores the username of the database which will be used by JDBC to connect to database.

- **JDBC_PASSWORD:**

  This attribute stores the password of the database which will be used by JDBC to connect to database.

Methods:

- **getConnection():**

  This method helps Module1 connect to database using JDBC.

- **writeFeatures():**

  This is a method that writes the high level features on the database for the Case Id with the Label. The Label can be 'MSU' for Michigan State University and 'UM' for University of Miami.

Design of Module2:

This module returns a set of Case Ids for the dental records which bares the features given in by the user. The features can be any non-dental feature like mortality tag, gender etc. At present this module takes mortality tag as the only feature. The Class Diagram for Module2 is as give below:

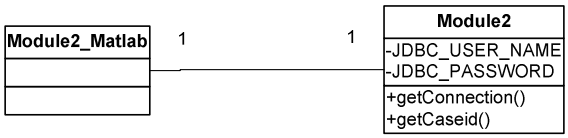


Figure 10: Class Diagram for Module2

Description of Class Diagram:

Module2_Matlab is a Matlab file which calls the methods from the Module2 class to get back the results.

Class: Module2_Matlab

This is a matlab file which makes an object out of Module2 class and uses it.

Class: Module2

This is the class which connects to database and returns an array of Case Ids of dental records which satisfy a feature set.

Attributes:

(Refer to attributes section of Class: Module1 in section 5.3.7.1)

Methods:

- **getConnection( ):**

  This method helps Module2 connect to database using JDBC.

- **getCaseid( ):**

  This method interacts with the database to get back array of Case Ids for the feature set given as input. The feature that can be mentioned in the latest version is only mortality tag. The Case Ids returned are all those which have the mortality tag that is given as input.

Design of Module3:

The main objective of Module3 is to get the high level features, Image of dental record and the preprocessing data of the dental record. This Class diagram of Module3 is as given below:

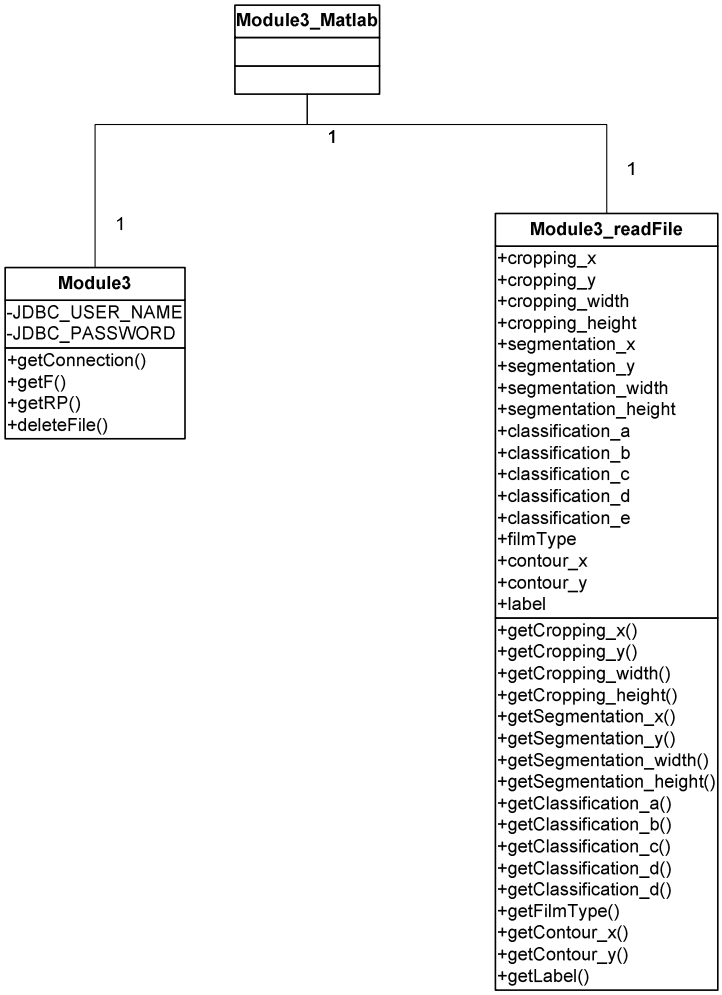


Figure 11: Class Diagram for Module3

Description of Class Diagram:

Module3_Matlab is a class in Matlab which uses the services provided by Module3 to get the High level features, image of dental record and preprocessing data. Module3_readFile reads the contents of the file to present the preprocessing data to the user.

Class: Module3_Matlab

This is a class in Matlab which uses Module3 and Module3_readFile to get the high level features, image of the dental record and preprocessing data.

Class: Module3

This is the class which reads the preprocessing data, gets the dental images and the high level features for a CaseId.

Attributes:

- **JDBC_USER_NAME:**

  This is the attribute which stores the username of the database.

- **JDBC_PASSWORD:**

  This is the attribute which stores the password of the database.

Methods:

- **getConnection():**

  This is the method which connects to the database server using Java Database Connectivity.

- **getF():**

  This is the method that returns the high level features for a caseid.

- **getRP():**

  This is the method that returns the dental record and the preprocessing text file

which contains the preprocessing data of the dental record.

- **deleteFile():**

  This method helps to delete the preprocessing text file after the preprocessing data is extracted.

Module3_readFile:

This class reads the preprocessing text file to get the values of the parameters needed by the bridge module.

Attributes:

- **cropping_x:**

  This attribute stores the x-coordinate of all the cropped films.

- **cropping_y:**

  This attribute stores the y-coordinate of all the cropped films.

- **cropping_width:**

  This attribute stores the width of all the cropped films.

- **cropping_height:**

  This attribute stores the height of all the cropped films.

- **segmentation_x:**

  This attribute stores the x-coordinate of all the segments from the films.

- **segmentation_y:**

  This attribute stores the y-coordinate of all the segments from the films.

- **segmentation_width:**

  This attribute stores the width of all the segments from the films.

- **segmentation_height:**

  This attribute stores the height of all the segments from the films.

- **classification_a:**

  This attribute stores the classification parameter named ‘a’.

- **classification_b:**

  This attribute stores the classification parameter named ‘b’.

- **classification_c:**

  This attribute stores the classification parameter named ‘c’.

- **classification_d:**

  This attribute stores the classification parameter named ‘d’.

- **classification_e:**

  This attribute stores the classification parameter named ‘e’.

- **filmType:**

  This attribute stores the type of the film. It can take values like bitewing, Periapical upper and Periapical down.

- **contour_x:**

  This attribute stores the x-coordinate of the contours of tooth.

- **contour_y:**

  This attribute stores the y-coordinate of the contours of tooth.

- **label:**

This attribute stores the label of the tooth.

Methods:

- **getCropping_x():**

  This method returns cropping_x attribute.

- **getCropping_y():**

  This method returns cropping_y attribute.

- **getCropping_width():**

  This method returns cropping_width attribute.

- **getCropping_height():**

  This method returns cropping_height attribute.

- **getSegmentation_x():**

  This method returns segmentation_x attribute.

- **getSegmentation_y():**

  This method returns segmentation_y attribute.

- **getSegmentation_width():**

  This method returns segmentation_width attribute.

- **getSegmentation_height():**

  This method returns segmentation_height attribute.

- **getClassification_a():**

  This method returns classification_a attribute.

- **getClassification_b():**

This method returns classification_b attribute.

- **getClassification_c():**

  This method returns classification_c attribute.

- **getClassification_d():**

  This method returns classification_d attribute.

- **getClassification_e():**

  This method returns classification_e attribute.

- **getFilmType():**

  This method returns filmType attribute.

- **getContour_x():**

  This method returns contour_x attribute.

- **getContour_y():**

  This method returns contour_y attribute.

- **getLabel():**

  This method returns label attribute.