Politehnica University of Bucharest

Software Project Management

Binarization Algorithm Module

Software Design Document

Coordinator:

Mihai Zaharescu

Team:

**Irina Talmaciu, gr 1272E**

**Gabriel Silaghi, gr 1272E**

**Teodora Mohai, gr 1272E**

***Ciulu Razvan, gr 1272E***

Contents

[Purpose 2](#_Toc371628332)

[Objectives 2](#_Toc371628333)

[Overview 2](#_Toc371628334)

[Data design 2](#_Toc371628335)

[Global data structures 2](#_Toc371628336)

[Linking 3](#_Toc371628337)

[Temporary 3](#_Toc371628338)

[File formats 3](#_Toc371628339)

[Database description 3](#_Toc371628340)

[Database structure diagram 3](#_Toc371628341)

[Table description 4](#_Toc371628342)

[Architectural design 5](#_Toc371628343)

[System architecture 5](#_Toc371628344)

[Architectural patterns 5](#_Toc371628345)

[Architecture diagram 5](#_Toc371628346)

[Description 5](#_Toc371628347)

[Implementation requirements 5](#_Toc371628348)

[Component interaction (Configuration items) 5](#_Toc371628349)

[User interface 5](#_Toc371628350)

[Flow chart 5](#_Toc371628351)

[Screen images 5](#_Toc371628352)

[Testing issues 5](#_Toc371628353)

[Critical components 5](#_Toc371628354)

[Alternatives 5](#_Toc371628355)

[Workdown Break Structure 5](#_Toc371628356)

[Gantt 5](#_Toc371628357)

# Purpose

This document intends to provide a detailed description of a binarization process applied in a software system which is intended to allow the interpretation of a color picture, conversion via a given algorithm and generation of an end result. The desired outcome is a black and white replica of the initial picture.

# Objectives

This project has included in its checklist the following:

* Research: search for binarization algorithms and techniques and build a data repository for them with characteristics
* Analysis: previously gathered findings are meant to produce a strength / weakness comparison which will be used to decide on one
* Decision: pick an algorithm/method based on previous step and present a valid argumentation
* Build documentation: construct the needed documents for the requirements/ design / test and implementation phases
* Implementation: build the software product corresponding to the previous documents
* Test: internal software product testing
* Presentation or Go-Live: run the software product on given input samples and present outputs

# Overview

A preliminary phase was completed with the realization of the Software Requirements Document which contains information around research, analysis and decision steps listed in the Objectives section.

In the following parts of this paper we will offer insight on the design phase of the software product. This will be related to data, architecture, user side or graphical interface, testing and project management tasks.

# Data design

When speaking about data we need to keep in mind several aspects such as: types of data/information, places where it is used, relationships between various fragments/elements of data, structures in which data is organized and flows that will be the paths of particular data elements through the software product body (code).

## Global data structures

In this section we will classify the data structures used throughout the code under their scope and or purpose:

* Basic: INT, FLOAT, STRING, ARRAY, LIST, etc.
* Image related: BufferedImage – image storing and handling format used in java to manipulate an image loaded into memory and perform operations on it;
* File related: BufferedReader – file storing and handling format used in java to manipulate a file loaded into memory and perform operations on it including read, write, update, generate, etc.
* Database related: DB connection – allowing the setup of a connection through sockets and other parameters to a local or network/internet shared database
  + - DB records – allowing access to, or capabilities to retrieve/delete/update information inside the DB
* GUI related: Frame, Buttons, Triggers and Events, Panels

## Linking

The data structures will be used in scopes corresponding to their capacity to maximize software productivity. Linking will be maintained error free and within predefined templates throughout the code.

## Temporary

We can name temporary all objects created in memory for use in instance operations such as file retrieval/push, database connection, image processing in memory.

## File formats

Image – we need to have the same file format both for the input file and for the output file. Since we are processing images the file format we decided to use will be images in a JPEG format (using the extension \*.jpg).

## Database description

In order to keep track of all the changes and images resulted from all the repetitive thresholdings we will use a simple database where we will store a history of all the image processing. The database will also help us to ensure the continuous flow of actions of the application.

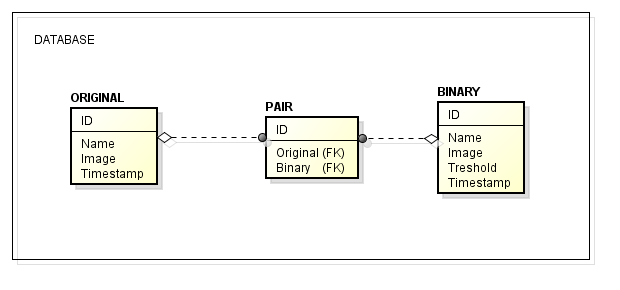
The implementation of the database will be realized by using the MySQL programming language. This choice was made based on the compatibility with de interface programming environment (Java).

## Database structure diagram

The database structure doesn’t need to be too complex to serve our purpose. A simple database with not more than two entities will be sufficient for implementing the application.

The first entity will describe and represent the original image and the second entity will represent the modified (binarized) image. For each original image we will have some corresponding processed images. Therefore we will have a one-to-many relationship between the two entities. In order to make a valid architecture we will use an intermediary entity which will store the pairs of images.

The database structure is represented by an ER diagram as we can see in Figure 1.

1. *Figure 1. ER Diagram of the database*

## Table description

According to the diagram described in section 4.f. the database will be formed out of three tables:

1. Originals – filled with all the original images selected by the user

|  |  |  |
| --- | --- | --- |
| **Originals** | | |
| **ID** | **PK** | Int NOT NULL AUTOINCREMENT |
| **Name** |  | Varchar(30) |
| **Image** |  | Blob |
| **Timestamp** |  | timestamp |

1. Pairs – filled with all the existing original image -binary image pairs

|  |  |  |
| --- | --- | --- |
| **Pairs** | | |
| **ID** | **PK** | Int NOT NULL AUTOINCREMENT |
| **Original** | **FK** | Int NOT NULL (Originals(ID)) |
| **Binary** | **FK** | Int NOT NULL (Binaries(ID)) |

1. Binaries – filled with all the images resulted after processing.

|  |  |  |
| --- | --- | --- |
| **Binaries** | | |
| **ID** | **PK** | Int NOT NULL AUTOINCREMENT |
| **Name** |  | Varchar(30) |
| **Image** |  | Blob |
| **Threshold** |  | Int |
| **Timestamp** |  | Timestamp |

# Architectural design

## System architecture

## Architectural patterns

## Architecture diagram

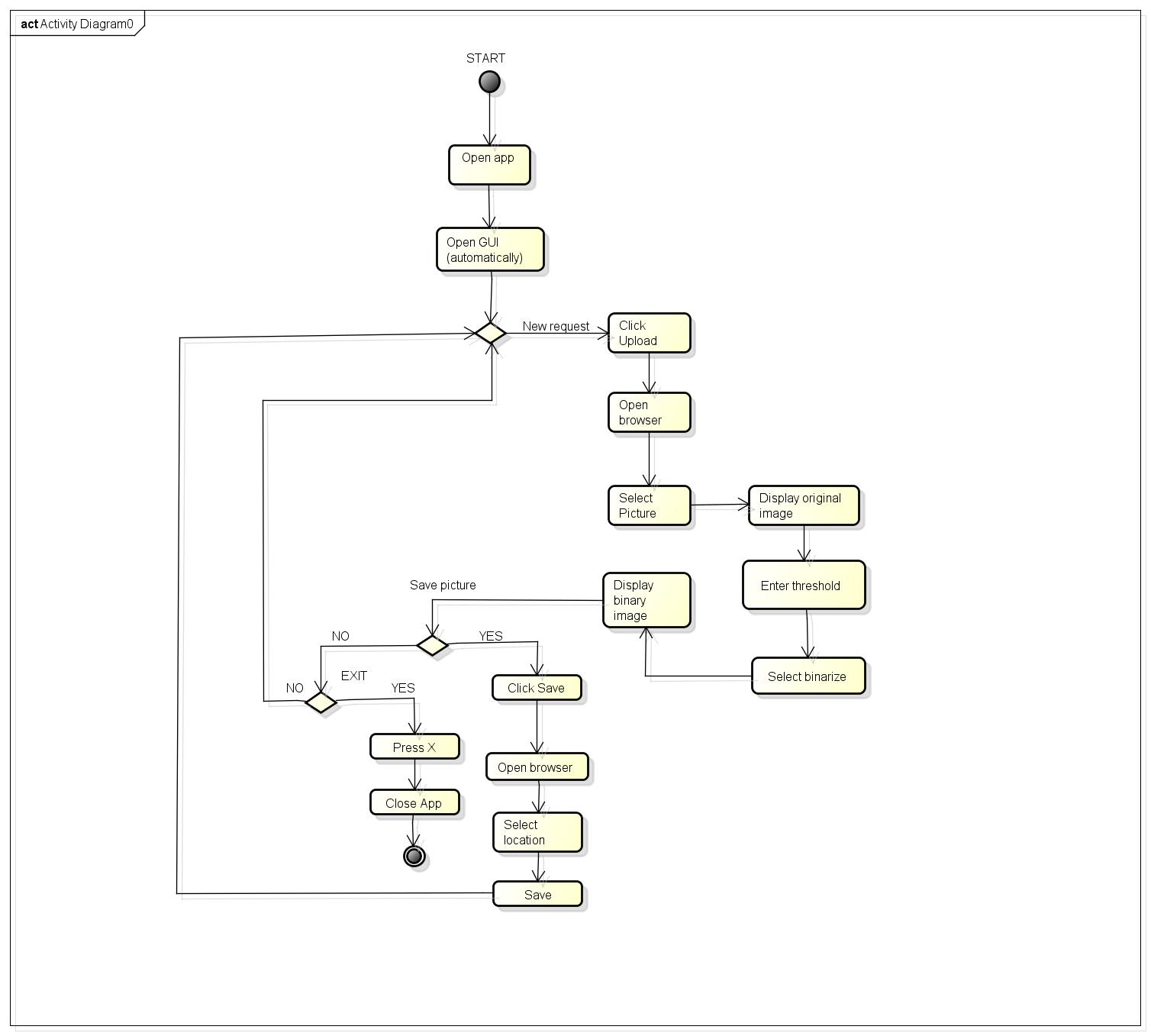
## Description

## Implementation requirements

## Component interaction (Configuration items)

# User interface

## Flow chart



## Screen images

# Testing issues

## Critical components

## Alternatives

# Workdown Break Structure

# Gantt