



Middle East Technical University
Department of Computer Engineering

Computer Engineering Design I
fall 2010

Initial Design Report

for

bookwiser augmented reality for libraries

The Ballmer Peak

Content

| | |
|--|----|
| 1 Introduction | 4 |
| 1.1 Problem Definition..... | 4 |
| 1.2 Purpose..... | 4 |
| 1.3 Scope..... | 5 |
| 1.4 Overview..... | 5 |
| 1.5 Definitions , Acronyms and Abbreviations..... | 6 |
| 1.6 References..... | 7 |
| 2 System Overview | 8 |
| 3 Design Considerations | 10 |
| 3.1 Design Assumptions, Dependencies and Constraints | 10 |
| 3.2 Design Goals and Guidelines | 13 |
| 4 Data Design | 15 |
| 4.1 Data Description | 15 |
| 4.2 Data Dictionary | 16 |
| 5 System Architecture | 20 |
| 5.1 Architectural Design | 20 |
| 5.2 Description of Components | 23 |
| 5.2.1 Object Recognizer Component..... | 23 |
| 5.2.1.1 Processing narrative | 23 |
| 5.2.1.2 Interface description | 23 |
| 5.2.1.3 Processing detail | 24 |
| 5.2.1.4 Dynamic behavior | 26 |
| 5.2.2 World Model Component..... | 32 |
| 5.2.2.1 Processing narrative | 32 |
| 5.2.2.2 Interface description | 32 |
| 5.2.2.3 Processing detail | 33 |
| 5.2.2.4 Dynamic behavior | 35 |
| 5.2.3 Graphic Component..... | 38 |
| 5.2.3.1 Processing narrative | 38 |
| 5.2.3.2 Interface description | 38 |
| 5.2.3.3 Processing detail | 38 |
| 5.2.3.4 Dynamic behavior | 40 |
| 5.3 Design Rationale..... | 42 |
| 6 User Interface Design | 43 |
| 6.1 Overview of User Interface | 43 |
| 6.2 Screen Images | 43 |

| | |
|---|-----------|
| 6.3 Screen Objects and Actions | 51 |
| 7 Libraries and Tools..... | 53 |
| 8 Time Planning (Gantt Chart)..... | 54 |
| 9 Conclusion..... | 55 |

1 Introduction

1.1 Problem Definition

Libraries are the most important places for the people who are reading books and doing researches; that is to say, people who want to learn something new usually go to the libraries.

However, this process is not so easy that people waste a lot of time while searching what they want. People have to spare considerably much time while searching for a book, or books that s/he can like in a genre that s/he wants.

Moreover, the same situation happens for the people who knows exactly what they want, namely search for a specific scientific magazine or specific book or even the books of a specific author.

Since the current databases of libraries lead to the user only the correct floor of the library and gives the correct bar-code of the shelf that the user is looking for, this waste of time is inevitable. Also, A user who wants to read a book but does not have a decision on which book to read has to take each book from the shelf, open each book and try to find some information that may help her/him to decide.

Making the user's travel in the library more efficiently by recognizing objects namely the shelves, the books and even the librarian, after that showing information about the library environment on user's screen using augmented reality concept is a perfect solution to these kind of problems. Bookwiser is the application that leads the user to the correct place in an reasonably rational time.

1.2 Purpose

This document includes initial design of Bookwiser Software Project.

The project uses augmented reality concept to make user get easily informed about the organization of the library, namely the books, content of the books and also interact with the librarian.

The project is essentially composed of three parts.

First part is the object recognition part which includes the shelf recognition, book recognition and librarian recognition by using object detection techniques.

The second part is the interaction with the library database from which all the information about a book is extracted.

The last part of the project is the GUI part which is going to show the modified video stream

augmented with information related to recognized objects on the camera, provide options for user to choose the flow of the application and make user's experience in the library more effective.

The intended audience is for this document is people who deal with further development of Bookwiser.

1.3 Scope

The project is named "Bookwiser", since the application seems to know everything about every book in the library, and helps user about books

Our project will be a augmented reality application for a library environment which works on mobile devices. When a user starts the application, Bookwiser first enables the video in the library. While the user is moving with the camera in the environment, Bookwiser will capture data from environment.

The user may choose to search a specific book or decide just walk and get informed about different kind of books in the library. In the first option the user gives the name of the book that he/she wants to reserve and Bookwiser will guide the user in order to find the correct shelf and the correct book. The other option is that the user has no idea about what he/she exactly wants , therefore Bookwiser will guide the user that while user is walking around the library, it will give information about the shelves and books that is in the predefined range of the camera.

When Bookwiser comes up with a book shelf belonging to a specific kind of books, it will detect the shelf by using object recognition in a predefined range. After the shelf detection & classification, the project will inform the user about the content of the books in the shelf.

When Bookwiser recognizes a book, it will show the rating of the book to the user using augmented reality and it will give detailed information to the user if the user wants. It will give user choices like rating the book or reserving the book.

Moreover, there will be librarians in the library for further questions and they will be detected using the predefined specifications. We think that people going to the library are mostly wasting their time for searching the correct place of a book or deciding which book to read. So that, Bookwiser very helpful to guide people in libraries.

1.4 Overview

The contents of this document consist of 9 basic parts:

- Introduction where the problem definition, scope, overview, definitions/abbreviations and references are explained,

- System Overview where the general description of the system is provided,
- Design Considerations where the special design issues, dependencies, constraints, goals and guidelines are noted,
- Data Design,
- System Architecture where the description of the program architecture is presented by describing each of components,
- User Interface where the user perspective of the project is explained,
- Libraries and Tools,
- Time Planning where the Gantt Charts are presented,
- Conclusion where the all parts concluded

1.5 Definitions and Abbreviations

Already-recognized Object: An object that is recognized when the previous frame is processed.

Augmented Reality: Term for a live direct or indirect view of a physical real-world environment whose elements are augmented by virtual computer-generated sensory input such as sound or graphics. For Bookwiser application, augmented reality represent the user graphical output that shows the recognized object properties.

Bookwiser: Name of the project.

Camera: A device that provides a video stream

Candidate Object: An assumption of boundary areas for possible images on an image

CPU: Hardware component also named as processor.

Filter: A sequence of image processing operations to enhance an image.

Frame: An image captured from the camera.

GPU: Graphics Processing Unit

GUI: Graphical User Interface

Image Processing: Image processing is any form of signal processing for which the input is

an image, such as a photograph or video frame. For Bookwiser application, image processing is the process of generating information from video stream captured by user camera.

IDR: Initial Design Report

Library: Real building where books or other kind of documents are stored for public use.

Object recognition: Object recognition is the process of retrieving identifying information about objects. Object recognition process is directly connected with image processing.

Object recognition range: The closeness of the camera to the real object which provides enough detail to recognize the object.

Video Stream: Video stream is the common name of Bookwiser's input and output format. It consists of stream of real time images.

1.6 References

1. IEEE Std 830-1998: IEEE Recommended Practice for Initial/Detailed Design Requirements
2. Class Diagrams, IBM,
<http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/>
3. Sequence Diagram, IBM
<http://www.ibm.com/developerworks/rational/library/3101.html>
4. Component Diagram, IBM
<http://www.ibm.com/developerworks/rational/library/dec04/bell/>
5. Data Flow Diagram, Wikipedia,
http://en.wikipedia.org/wiki/Data_flow_diagram

2 System Overview

Bookwiser is a real-time image processing and augmented reality application for libraries. Its main purpose can be described as assisting the library user in his/her library trip via a display device. To be more precise, Bookwiser provides useful information about library objects which are books, shelves and librarians.

To perform its task, Bookwiser processes the video captured by the camera frame by frame, generates the object properties, retrieve context information for these objects from external library database, manipulates the video according to these contextual and visual properties of objects and displays the final modified video frames via a display device.

Since the information shown to the user is purely based on instant camera view, performance is a crucial requirement for Bookwiser. What indicates the performance of the system is the execution times of image processing and manipulation actions. It is better to say that, Bookwiser can be classified as an augmented reality application more than a database driven system.

The core of its design consists of image manipulation and object recognition. On top of this core, there is the user interface which complements Bookwiser's functionality from the aspects of usability and attractiveness.

Except from the internal object database, which is planned not to be an actual database system, library database operations to get context information are almost completely out of the scope of the design for two reasons.

- First reason can be described as the aim of the system, which is recognizing objects in real time. Once the objects are recognized and identified by Bookwiser, desired information can be retrieved by simply querying it.
- Second reason is the obvious system's independence of the database design. It is nonsense to design an internal context database while there is already a well-established library database.

From the aspect of software design, Bookwiser is a usable, reliable and crucially fast object recognition and augmented reality visualization system by which the user can be able to freely walk around the library without spending time for examining the books by himself/herself.

Block diagram of the system is given below:

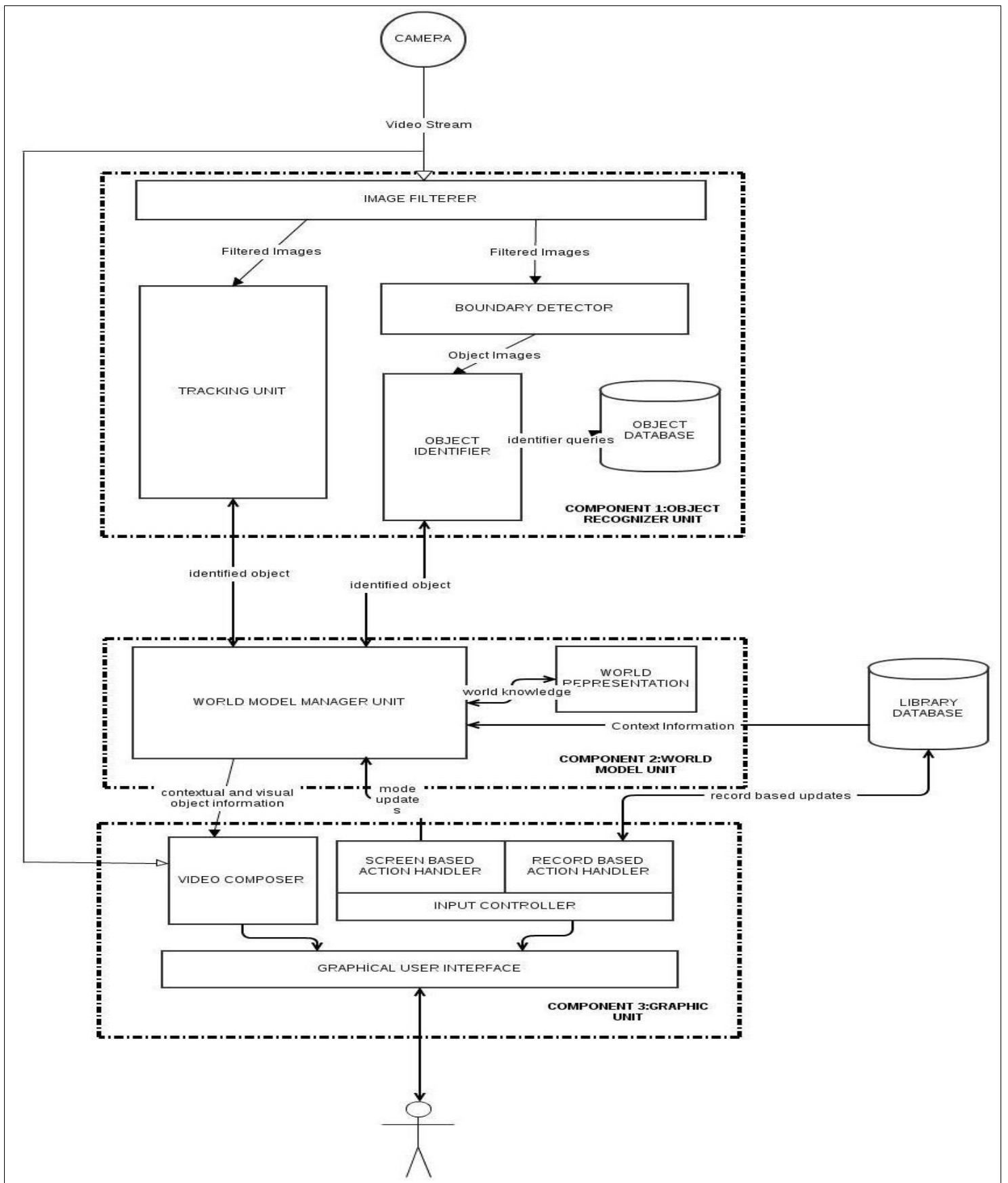


Diagram 2.1 – Block Diagram

3 Design Considerations

In this section, special design issues that should be considered during design and development are stated.

3.1 Design Assumptions, Dependencies and Considerations

3.1.1 Hardware related Assumptions, Dependencies and Considerations

- Bookwiser is designed to work on a portable device , including a camera, a display screen, RAM , hard-disk and processor units , and a graphics card.
- The system is assumed to a work on a device with a minimum of 2.0 GHz Intel processor and a minimum of 500 core GPU, no shared memory.
- The system is assumed to work on a device with a minimum of 20GB hard-disk and 1024Mb RAM.
- Bookwiser is assumed to operate on a device that is able to use wireless network communication. Therefore the device should have a wireless communication device and should be able to connect to a local network or the Internet.
- It is assumed that the system needs a minimum of 5.0 mega-pixels camera to feed the system with images with necessary quality and detail to do object recognition.
- Bookwiser will not use external GPS or any other satellite/GSM based navigation systems or devices to locate or recognize books / shelves.
- User will interact with the system via a keyboard.

3.1.2 Software related Assumptions, Dependencies and Considerations

- The system that Bookwiser will operate on should have a minimum version of Ubuntu 9.0 or Windows XP installed on it. The system should be suitable for QT and OpenCV platforms.
- Bookwiser needs to communicate with the library database , therefore it is assumed that the library database is open to Bookwiser and the library database is able to share information via Internet or a local network. If there's a local network, also the library should provide access to the network via wireless.
- There may be a need for creation of an extra table or a change in the existing table in the library database for user registration and related user information if the library

database doesn't suit application needs. Otherwise, Bookwiser should not make any changes on the structure of the library database in any cases.

- Bookwiser uses the library's own database to show data about recognized books. It is assumed that book information exists in the library database is correct and Bookwiser doesn't have to check the correctness of the data. The application itself is not responsible for book information organization in the database.

3.1.3 Performance related Assumptions, Dependencies and Considerations

- Bookwiser needs to assist the user during its travel in the library therefore the system should respond to the user immediately. The object recognition work should be done in near real-time and the overhead of this operation should be maximum 0.8-0.9 seconds on the worst case.

- To get the information about recognized object from the library the system needs to connect the library, because a wireless connection may take time, after an object is recognized, the overhead of gathering information about the object can take longer. But in the worst case it should take maximum 5 seconds on the worst case to gather the information from library database and to show the user via the graphical user interface.

- Bookwiser should have the capacity to serve all the visitors in a library and because each device that is provided for different users own a different instance of the system scalability is not a problem, on the other hand, because the library database may not be able to serve all Bookwiser users at the same time, connections to the database should be made under the consideration of limits of the server of the library database.

- Bookwiser may not be able to detect objects from long distances, but the system should be designed to at least recognize objects from the distances listed below:

- Books : 2-2,5 meters.
- Shelves : 4-6 meters.
- Librarians : 3-5 meters.

3.1.4 Interface related Assumptions, Dependencies and Considerations

- The interface to interact with the camera is provided by OpenCV library that used by the system. To gather information from the camera, Bookwiser will use OpenCV's camera methods and classes.

- Bookwiser uses a library's own database to gather user, book and shelf data. The interface to communicate with the library database is dependent on library database's architecture. Therefore the system should be designed flexible enough to be able to interact with different library databases.

3.1.5 Safety & Security related Assumptions, Dependencies and Considerations

- The devices which Bookwiser is run on should be tested for health issues, the screen and the device components should obey the standards.
- The library database manipulations done by Bookwiser are limited by the library constraints, book information addition/deletion/manipulation or access to other users' information is not made by Bookwiser to protect library database from possible misuse.
- Users' passwords and detailed information should not be shared with other users to protect the system from illegal access.
- Before unrecoverable operations users should be asked if they are sure about the operations.
- The library, according to its own rules, may block a user's access to the system or just book reservation functionality due to misuse of the system. The decision is library's and only responsibility of Bookwiser is to inform the user via graphical interface.

3.1.6 Standards

- It is assumed that the library have adequate light sources around the objects
- It is assumed that shelves in the library are equally spaced with a minimum of 1 meters and they are placed parallel to each other.
- It is assumed that shelves in the library are standard and they have the same height/width and shape. Shelves should have a color that can be differentiated from the library environment and special marks on the shelves should be apparent and proper enough to classify them.
- It is assumed that books are placed on the shelves with their front side facing the camera and the user. The books are assumed to be placed with equal spaces between them, a minimum of 10 centimeters. Books should have a color that can be differentiated from the library environment and special marks on the books should be apparent and proper enough to classify them.
- Librarians should wear a standard t-shirt with the same color and a specific apparent shape on top-left of their t-shirts` front and back sides, making them recognizable.
- The marks, values or features to recognize and classify books, shelves and librarians should be predefined at the object recognition database. **Bookwiser** will only recognize and classify objects with predefined marks , shapes and colors.

3.2 Design Goals and Guidelines

3.2.1 Usability & Real-time processing:

Bookwiser is a system that assists users actively during their tour in the library. Because the user needs to be responded immediately during their tour, the system should react the changes on the screen or inputs of user via graphical user interface without losing time. Otherwise the system can not be usable for library visitors.

This requirement brings the need that all the algorithms and structures in the system should be implemented to work very efficiently. The object recognition work should be done in near real-time and the overhead of this operation should be maximum 0.8-0.9 seconds on the worst case.

3.2.2 Minimalism:

Bookwiser is a project with a main goal of helping its users; therefore, the user interface of the system should be implemented in a way that it should not bother the user during his/her travel in the library. The graphical elements on the screen that shown as a part of the augmented reality concept should not be blocking the user's view , or disturbing the user. These graphical elements should be designed to be minimalistic and should look natural. User should always be able see the frames coming from the camera in a big part of the screen so that user is always able to move or see the objects around him/her.

3.2.3 The KISS Principle :

All the components , data structures and methods in the design should be implemented in the simplest way to serve the user the better. The main aim of Bookwiser is to create shortcuts for everyday procedures of a library user, so features like getting book information, registering a book etc. should do the maximum work with the least effort from the user. Use of different features of the system should be very simple for the users to use.

The project should not make it more difficult to travel in the library rather that making it much more simple for users.

3.2.4 Don't Repeat Yourself Principle :

To avoid repeats of parts of the system, the system is designed to use some core parts in different sub-components of each components, using an object oriented approach. Developers should stick to this property of the design and avoid duplicate system parts or data structures to increase the performance of Bookwiser.

3.2.5 Principle of Good Enough :

Bookwiser project is designed with a strong consideration of principle of good enough.

Components and functionalities of all system parts are designed to be simple and they can be seen as cores. These cores are suitable for further developments, or additions of extra functionalities. Developers should obey this principle to make the system suitable for future extensions.

4 Data Design

In this section, data elements are described and defined briefly.

4.1 Data Description

Our system Bookwiser, is transformed into files that contains classes. There are mainly four data units in our system in a data description manner. These units are Input Controller, Authentication, Graphical User Interface Unit and Object Recognizer Unit. In addition to these, there are two libraries and two external devices, a Camera and a keyboard, that Bookwiser is in a relationship with.

Library Database:

First database is the Library Database and it is not directly related to our system, because, this is the database where all book, shelf and user related, “object recognition independent” information are kept in and this database is not a part of the design. Organizing this database is not a responsibility of the application, instead this external database is only used to gather information.

Internal Object Database:

The second library is the main library that Bookwiser uses, which is Internal Object Database. The database keeps features for object recognition purposes. This database contains:

- KeyPoints as CvSurfPoint classes of OpenCV ,
- descriptors as floats,
- id fields as integers
- type fields as integers

WorldRepresentation:

This data structure is created at the start of the application and updated dynamically while the processing continues. What WorldRepresentation does is actually spine of the Bookwiser. Firstly it is composed of two tables StateChart and ObjectTable. ObjectTable does not only contains the objects coming from the frames that camera captures but also the objects created after the context information matching is processed with Library Database. Also, WorlRepresentation is updated by the information coming from Input Controller at each time. So we have a dynamic data structure while the application continues to process.

Other Components:

Other components of the system, Input Controller, Authentication, Object Recognizer and Graphical User Interface Unit are stored in the portable device that Bookwiser will use. The components are organized by the order of data flow and processed accordingly and we explained in the Section 5.

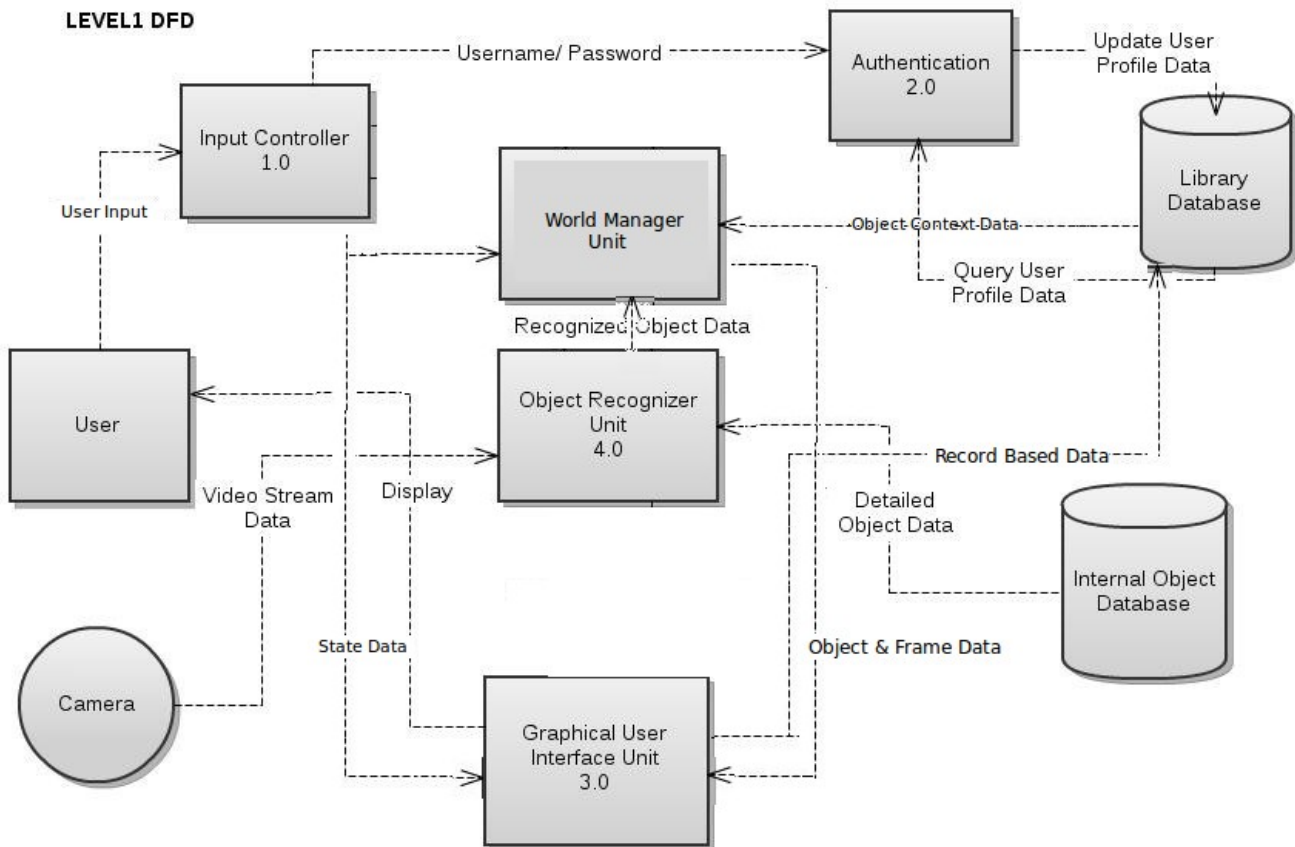


Diagram 4.1 – Data Flow Diagram

4.2 Data Dictionary

Since our approach in the Bookwiser is object-oriented, related data dictionary is composed of data system entities, which are data classes and their related methods. The whole entities and instances are separated according to component which belongs to, in an alphabetic order.

4.2.1 System Entities

4.2.1.1 Inherited Data Types

cvSurfPoint : keyPoint for SurfFeatures inherited from OpenCV library.

IpImage: a class for representing images, inherited from OpenCV library.

4.2.1.2 Object Recognizer Component

Feature: Data Class, used to implement Feature object.

Object: Data Class, used to implement a generic class for the three kind of specific objects: Book, Shelf and Librarian.

ObjectList: Data class, used to handle all of the objects.

Position: Data class, used to implement the Position object.

4.2.1.3 World Model Component

Book: Data class, used to implement a book object.

Librarian: Data class, used to implement the librarian object.

ObjectTable: Data class, used to hold the all objects of the Object class.

Shelf: Data class, used to implement a shelf object.

State: Data class, used to implement the state object that a state stands for the mode of the system .

StateChart: Data class, used to hold all states of the current object.

4.2.1.4 Graphic Component

VideoComposer: Data class, used to implement video frames into the object classes.

4.2.2 System Methods

4.2.2.1 Object Recognizer Component

addFeature(Feature): Object class method, used to add new feature to the current object.

addObj(): ObjectList class method, used to add a new object to the object table.

delFeature(int): Object class method, used to delete a feature given id as a parameter of the

current object.

getFeatures(): Object class method, used to return all of the features of the current object.

getLocation(): Object class method, used to return the location of the current object.

removeObj(id): ObjectList class method, used to remove a object given by the id parameter from the object table.

setLocation(Position): Object class method, used to change the position of the current object.

setPosition(): Position class method, used to change the position of the current object.

size(): ObjectList class method, used to return the size of object table in order to find out the number of objects located in the object table.

4.2.2.2 World Model Component

getAuthor(): Book class method, used to return the name of the author of the current Book object.

getId(): Shelf, Book and Librarian classes method, used to return id of the current object.

getName(): Book class method, used to return the name of the current Book object.

getRating(): Book class method, used to return the current rating of the Book object.

getState(): StateChart class method, used to return the current state of the system.

getType(): State class method, used to return of the type of the current State object.

insertObject(): ObjectTable class method, used to add a new object given as a parameter to the object table.

listObject(Object): ObjectTable class method, used to list the object given as a parameter.

removeState(State): StateChart clas method, used to remove the state object given as parameter from the StateChart class.

setId(id): Book, Shelf and Librarian classes method, used to change the id of the current object.

setName(Name): Book class method, used to implement any change of the Book object.

setState(State): StateChart class method, used to change the current state of the system by the State object given as parameter.

setType(type): State class method, used to change the type of the current State object.

setRating(rating): Book class method, used to change the rating of the current Book object.

updateObject(Object): ObjectTable class method, used to update an object given as parameter in the ObjectTable class.

4.2.2.3 Graphic Component

composeVideo(Object, float): VideoComposer class method, use to implement video frames and transfer to the object classes.

5 System Architecture

In this section a general description of the system architecture is given.

5.1 Architectural Design

From the most basic view, Bookwiser has a sequential way of operating. What it does is simply processing an image, retrieving some info from image, enriching this graphical information with some extra contextual information, combining them together in image and showing the reconstructed image to the user.

This operating style is reflected onto the design as three components of the system:

- Object recognizer to process images and create abstract object models about the library world.
- World model unit to compose a world view inside Bookwiser .
- And finally graphic unit to convert this abstract view to a graphical view for user. Functionally, world model units needs object recognition unit to operate, graphic unit needs world model unit to operate.

On the other hand, this style of abstraction -in its pure form- could not answer the performance goals of the system. To achieve timing goals which described in the section 3, it should be ensured that the system performs in predefined time limits. Performing this in a sequential flow of events is unrealistic.

Thus, three main components of Bookwiser (object recognizer unit, world model unit, graphic unit) should be designed as different threads which runs parallel and collaborates with each other. Three components performs their tasks to achieve timing goals of the system with synchronization.

Graphic unit has a higher priority than other two components since the main function of the Bookwiser is to answer the needs of user quickly. Yet it is logical to allow other two units to occupy the process time, because those two components have more processing weight.

While graphic units uses world model unit as a source of information, object recognizer unit and world model unit proceed collaboratively to create a representative picture of world. They identify the current world state one object at a time. In this way, Bookwiser can perform in its time limits. According to world state and this time limits, world model unit arranges the tasks of object recognizer unit. It can restrict the object recognizer unit with certain tasks or it can assist the object recognizer with world state information.

These three components are shown in the below component diagram:

Component Diagram

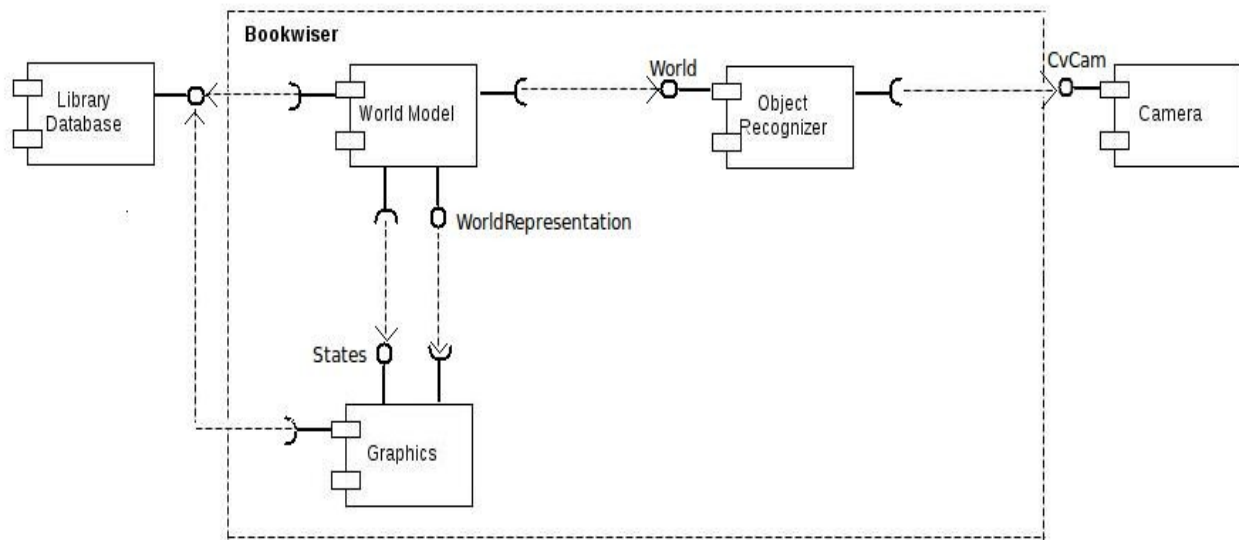


Diagram 5.1 – Component Diagram

A brief explanation of the components are given:

Object Recognizer Component:

This unit operates directly on the digital image sequences captured by the camera. It uses its own database to define objects. After every identification of a particular object-either a re-identification with tracking or discovery of a new object- it sends this information to world model unit and accordingly, it is assigned another identification task by the world model unit. Its functionality can be particularly or completely cut off according to timing constraints or user events.

World Model Component:

World model unit performs management tasks. All information gathered from the outside sources-camera, library database and the user- comes to the unit, is classified and used to produce an abstraction of world. It controls the working mechanism of object recognizer and serves as a knowledge base to the graphic unit. The world representation sub-component can be seen as a data table which arranged according to the object classifications and user inputs which defines the state of the Bookwiser. The manager sub-component is the interface of the world representation.

Graphic Component:

Graphic unit is responsible form every action which considers the user. These actions are two-way. Giving information to the user and getting information from him/her. Giving

information means displaying the camera view with augmented contextual information. Video composer is responsible for this task. Getting information is handled by input controller. User inputs are classified and sent to related components. Note that user inputs are not processed by the graphic unit. The graphic units functionality is restricted with maintaining the connection between user and the Bookwiser.

5.2 Description of Components

5.2.1 Object Recognizer Component

In this section , an explanation for the Object Recognizer Component and the related class and sequence diagrams are provided.

5.2.1.1 Processing Narrative for Object Recognizer Component

Object Recognizer Component can be described as the eye-brain channel of the Bookwiser system. All the object recognition, tracking and classification process is completed inside this component.

The component receives frames from the camera, filters the frame for a better quality image and operates on this image.

The component first finds assumptions for borders of possible objects on the scene. Using this assumptions , searches related areas of the image for SURF features. When an object is recognized, adds the object to a global recognized object list so that other components can use this available information to progress on their own duties.

Another responsibility of this component is to track already-recognized objects in the new frame. When an object is recognized in a frame, on the next frame the object's position can be changed or the object can be completely out of the scene. The component therefore decides to update the position of the object instance on the global object list or remove the object from the recognized object list.

This component has the heaviest load of the system and therefore is the most detailed component of the Bookwiser.

5.2.1.2 Interface Description for Object Recognizer Component

Generally, it can be said that the component has relations with two main interfaces.

- As an input interface, the component uses OpenCV Library's CvCam class to interact with the camera and to get sequences of frames. The detail of this interface will be given in the Detailed Design section.
- Output interface for the component is named as World. Since the component gives a description of the scene which the user is currently looking at to the system, using this interface, the component shares a global recognized object list and the current frame sent from the camera with the other components. Current frame and global object list , together implements the World interface for the use of other parts of the system.

5.2.1.3 Processing Detail for Object Recognizer Component

Object Recognizer Component describes the user's view to other components of the system.

- When a new frame is supplied by the camera , RecognitionHandler creates a sequence of filters to enhance the input image. With different FilterHandler instances, the image is enhanced for future operations.
- RecognitionHandler then calls the ObjectTracker . ObjectTracker looks at the ObjectList and if any recognized Object instances exist from the previous frame, ObjectTracker tries to find the new positions of objects in the frame.
- At this point ObjectTracker may find out that an object is no longer in the frame, than it removes the related Object instance from the ObjectList . If not so, then ObjectTracker updates the position information of the related Object instance in the ObjectList.
- When tracking operation is done, RecognitionHandler starts for a search of new objects in the frame.
- First BoundaryFinder is called to find assumptions for borders of book, shelf and librarian objects. BoundaryFinder may decide to remove some candidate objects it has found if it detects that the candidate object is a duplicate of one of already-recognized objects.
- The list of candidate objects than is used by ObjectRecognizer to find if they really are one of the objects that in the object recognition database.
- If it decides that the candidate object is one of the known objects, than it prepares an Object instance and adds it to ObjectList . If thats not the situation than the candidate object is deleted.
- After then RecognitionHandler asks for a new frame from the camera and the whole process starts again.

A more detailed explanation of the flow of the component and the details for the methods and algorithms of the component are left for the Detailed Design section.

The relationships between mentioned classes can be seen in the Class Diagram for Object Recognizer Component below.

Class Diagram - Object Recognizer Component

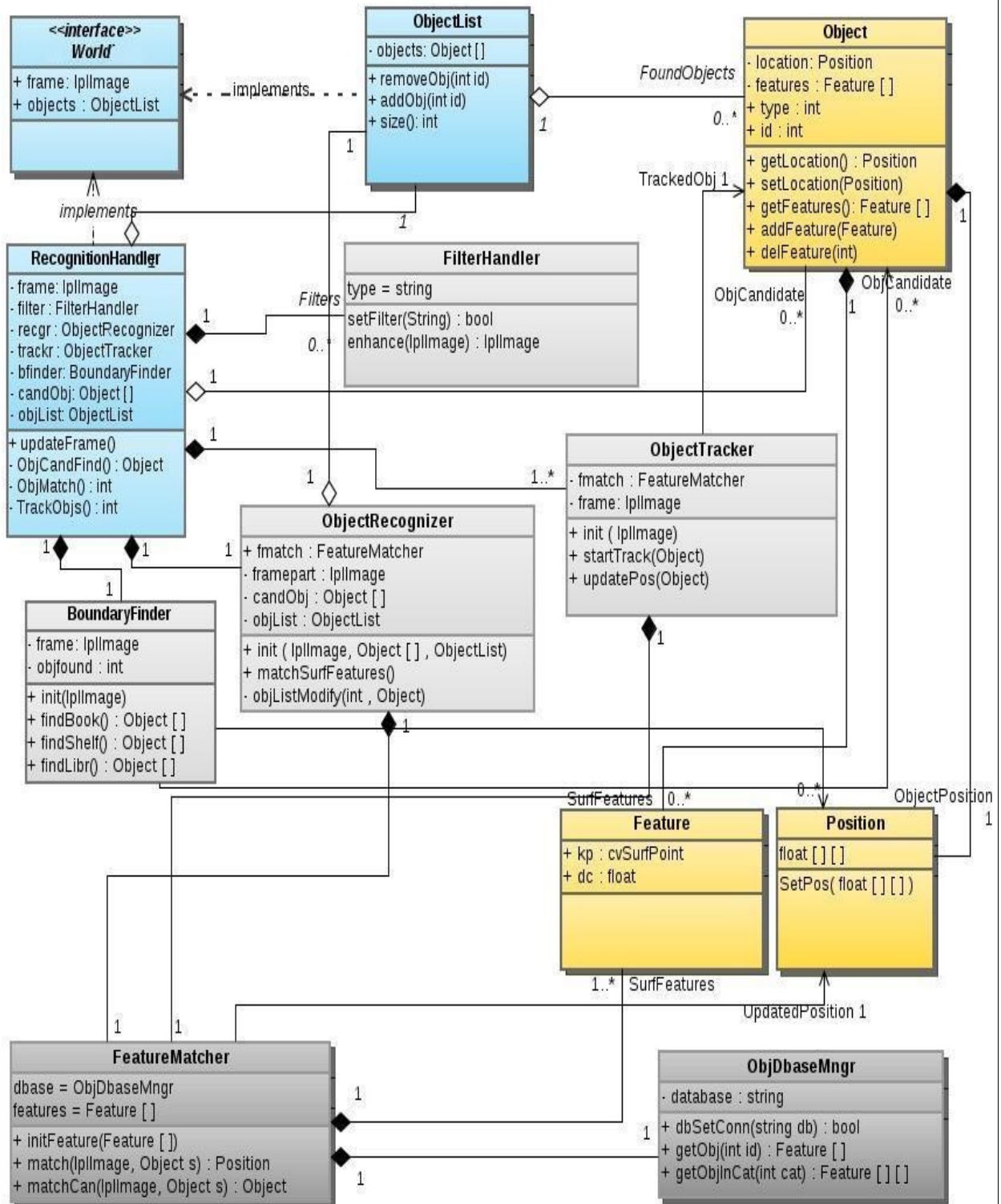


Diagram 5.2 – Class Diagram

5.2.1.4 Dynamic Behavior of Object Recognizer Component

Image Filtering:

When the RecognitionHandler gets a new frame from the camera , first image filtering is done.

- RecognitionHandler creates a FilterHandler instance and initializes it with a call to `setFilter()` method.
- If the initialization is successful then the frame is sent to the FilterHandler via the `enhance(IplImage)` method.
- After the enhancement operations image is returned back to the RecognitionHandler.
- The sequence is repeated for all filters defined by RecognitionHandler

Detailed information of filters that will be used in this part will be given in the Detailed Design section.

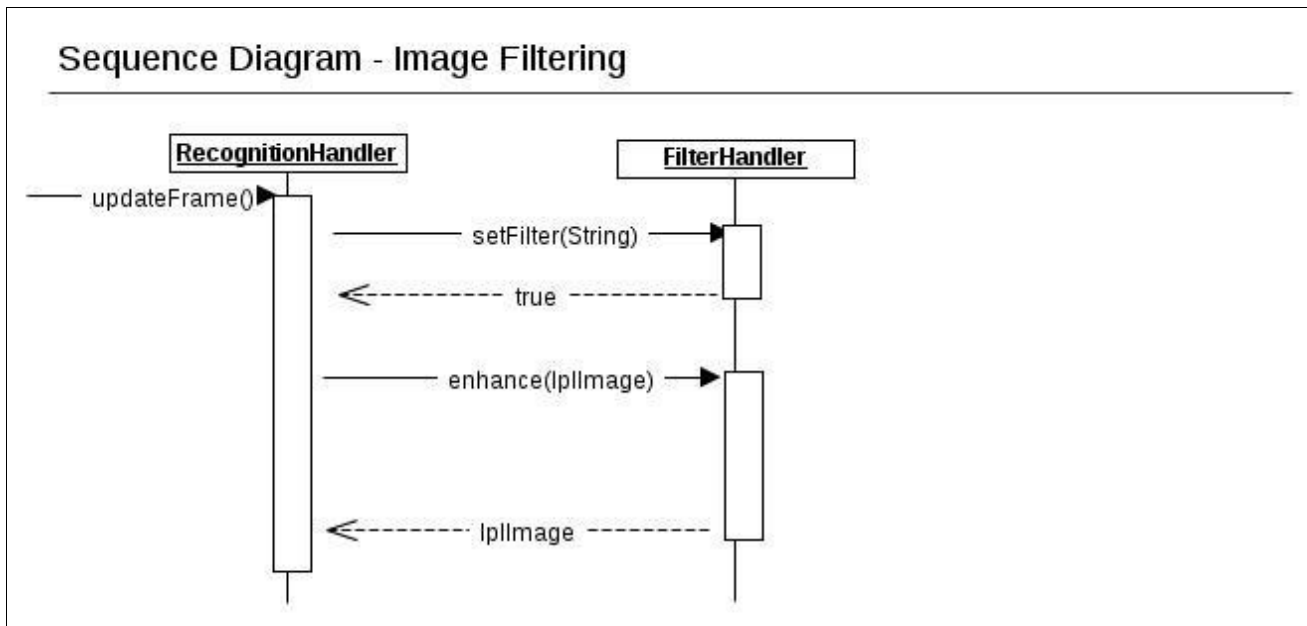


Diagram 5.3 – Sequence Diagram

Object Tracking :

After the filtering operations are done, object tracking is activated.

- RecognitionHandler , for each already-recognized object initializes an ObjectTracker instance with init(IplImage) method.
- After the initialization is done, startTrack(Object) starts the tracking operations. With getFeatures() method.
- ObjectTracker gets features from the related Object instance.
- ObjectTracker initializes a FeatureMatcher instance with a call to initFeature(Feature[]) method
- During initialization FeatureMatcher initializes a ObjDbaseMngr via a dbSetConn(String) call for object database operations.
- With a call to match(IplImage,Object) function, FeatureMatcher requests object features from the ObjDbaseMngr with getObj(int) and returned features are used to identify the position and orientation of the object.
- If it is not null , the new returned Position is used to update the Object instance's Position element with setLocation(Position) call.
- If the returned Position is null then the Object is deleted from the ObjectList with a call to removeObj(int).

Tracking algorithms that used in this part will be explained in the Detailed Design section.

Sequence Diagram - Object Tracking

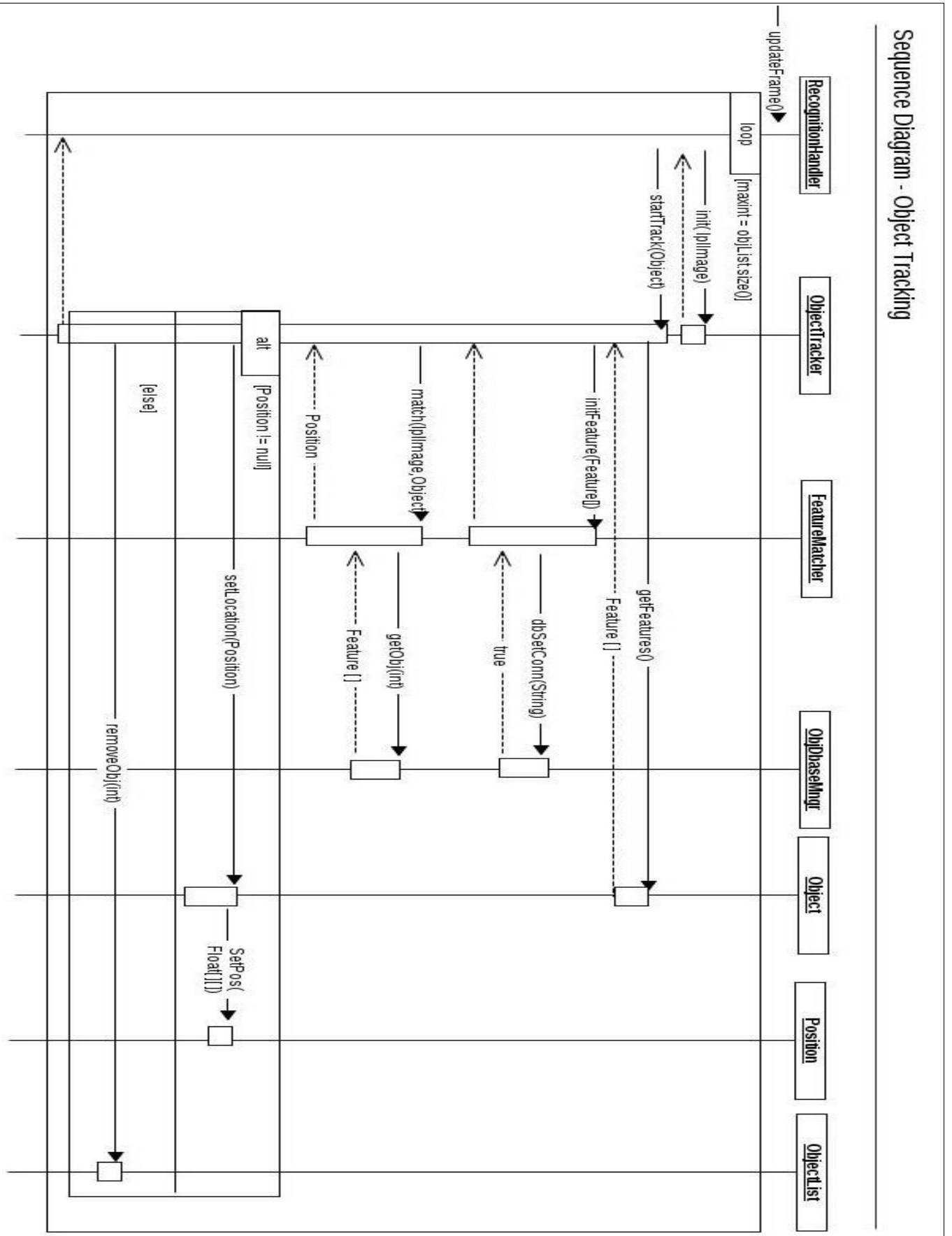


Diagram 5.4 – Sequence Diagram

Candidate Object Boundary Finding:

After Object Tracking is done, a search for new objects is started.

- RecognitionHandler initializes BoundaryFinder with `init(IplImage)` call.
- There are 3 methods that starts BoundaryFinder to search for candidate objects : `FindBook()` , `FindShelf()` and `FindLibr()` . The mentioned methods search for possible boundaries for books, shelves and librarians respectively and each method applies a different algorithm for possible boundaries of objects.
- For each found possible boundary a candidate Object is set .
- A list of found candidate objects are returned to the RecognitionHandler.

Algorithms for finding possible boundaries used in this part will be explained in the Detailed Design section.

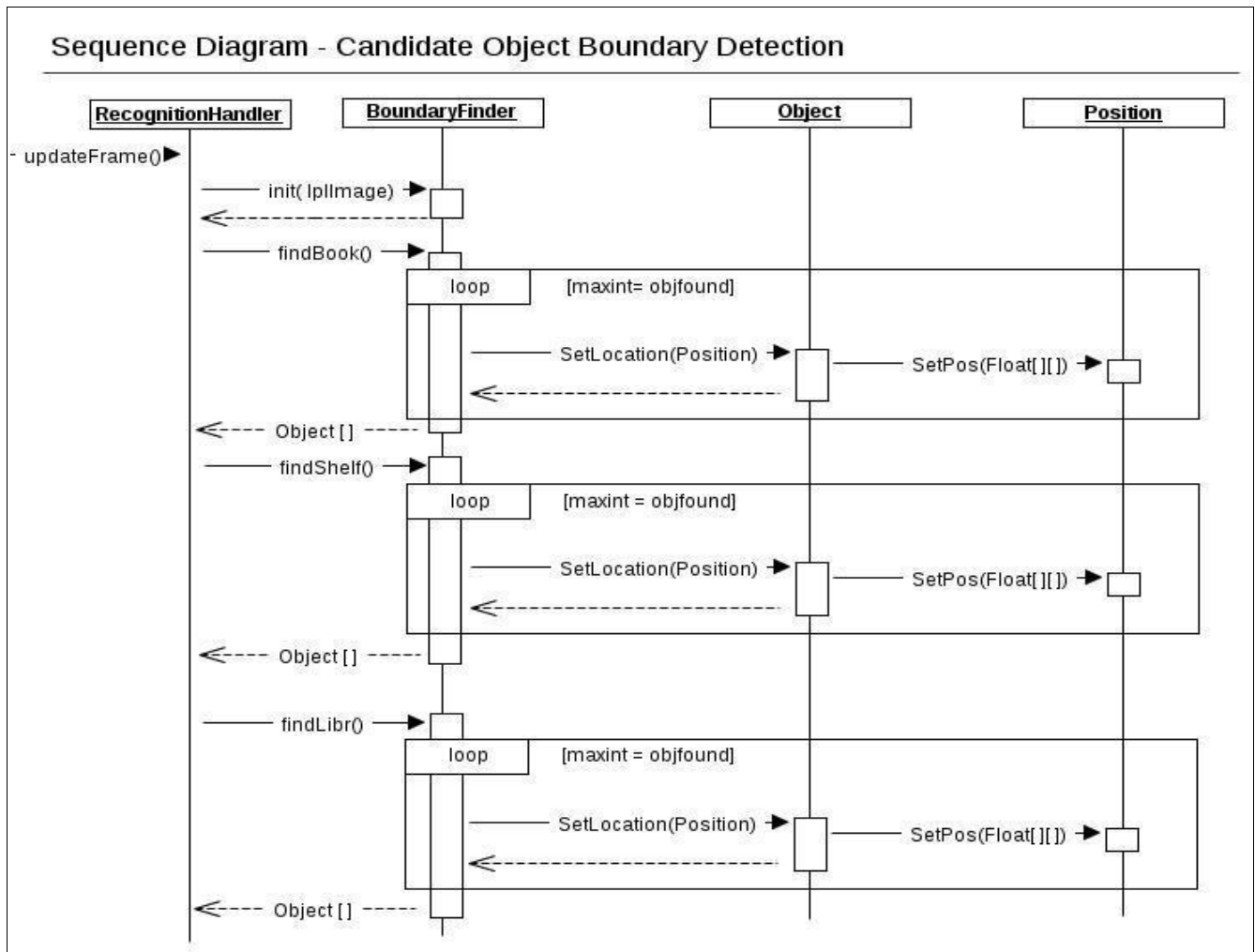


Diagram 5.5 – Sequence Diagram

Object Recognition:

- After candidate Objects are found, ObjectRecognizer is initialized by RecognitionHandler with a call to the method `init(IplImage, Object[], ObjectList)` .
- `MatchSurfFeatures()` method for each candidate Object starts a recognition process.
- ObjectRecognizer initializes a FeatureMatcher for each candidate object with `initFeature(Feature[])` call.
- FeatureMatcher sets a database manager, ObjDbaseMngr with `dbSetConn(string)` call.
- When these operations are done successfully , `matchCan(IplImage, Object)` function makes the FeatureMatcher compare features of the candidate Object with features in the object database after getting them from database with a call to the `getObjInCat(int)` method of ObjDbaseMngr .
- The matched candidate Object, is filled with known features with the `addFeature(Feature)` call and its Position is set via `SetLocation(Position)` method.
- After the candidate Object is transformed into an Object the Object is sent back to ObjectRecognizer .
- ObjectRecognizer adds the Object to ObjectList with `addObject(int)` call.
- If the candidate Object is not matched with any object records in the database, the candidate Object is deleted.

Feature matching algorithms will be explained in detail in the Detailed Design section.

Sequence Diagram - Object Recognition

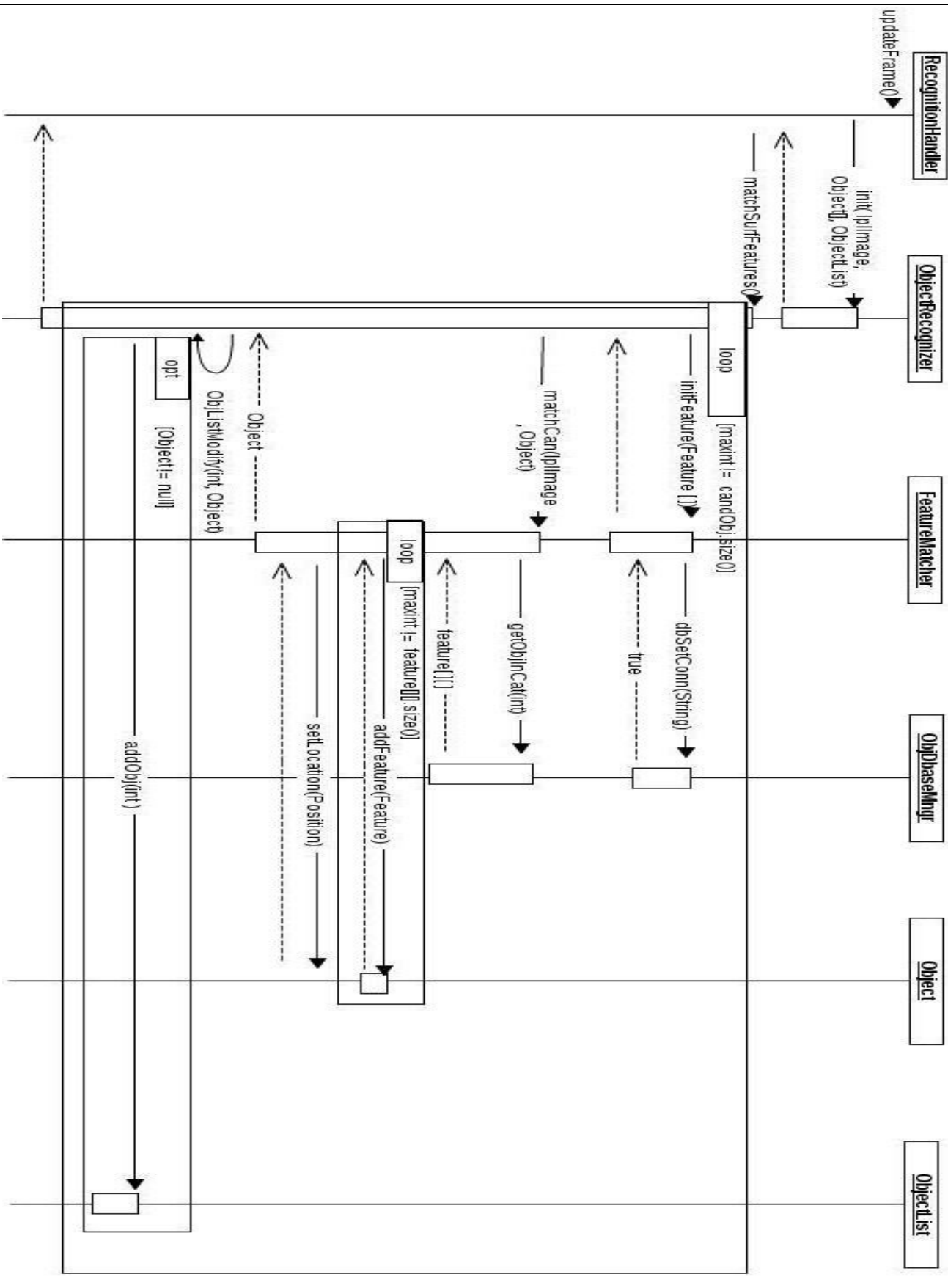


Diagram 5.6 – Sequence Diagram

5.2.2 World Model Component

In this section , an explanation for the World Model Component and the related class and sequence diagrams are provided.

5.2.2.1 Processing Narrative for World Model Component

World Model Unit is the main system component in the Bookwiser. World Model Unit is the system that collects all information from all components and sum up them to the user by the Graphic Unit because it connects the Object Recognizer Unit and Graphic Unit together.

World Model Unit is mainly the spine of the Bookwiser that all the components and data flow are controlled in this unit. All the information that flows in the application from the user to the user again should interact with this component.

The object information comes from the Object Recognizer Component. All the information about the detected objects like books, shelves and the librarian, are controlled in this component. The objects detected by the Object Recognizer Component, comes to the World Model Component. Now, this information has no meaning to the application because the recognized objects should be processed and filled with related information. For this purpose, World Model Component retrieves context information from the Library Database. Using the existing information in the database, World Model Component fills in the information fields of recognized objects . The object detection part is actually completed in this unit in this manner. After these operations , Bookwiser will show the related information to the user by Graphic Component.

On the other hand this component also keeps track of user-intercation based state changes. Different states of different user choices are stored and processed via this unit. According to different inputs from the user, the output that will be sent to the Graphic Component is manipulated in this component.

As a conclusion it can be said that this component is the brain of the Bookwiser system.

5.2.2.2 Interface Description for World Model Component

The interfaces in the World Model Component is not related with the user directly. Instead the two input interfaces of the component interacts with the Object Recognizer Component and the Graphic Component , and the output interface only interacts with the Graphic Component.

- The interface “World” is the input interface for interactions with Object Recognizer Component.
- The interface “States” is the input interface for interactions with Graphic Component.
- Moreover, “WorldRepresentation” is the output interface to the Graphic Component.

5.2.2.3 Processing Detail for World Model Component

An algorithmic description for the World Model Component as follows:

- 1) The information comes from the Object Recognizer Component to the World Model Manager Component.
- 2) After, "setConn" function is called and returned by true, Context Information coming from the Library Database is brought to the ObjectManager class in order to be combined with the information coming from the Object Recognizer Component.
- 3) Step2 is repeated whenever the information coming from Object Recognizer Component is updated.
- 4) Mode updates information comes from the Graphic Component to the World Model Manager Component.
- 5) Step4 is repeated when the user changes the state of the system or select one of the modes by the input controller such as Search Mode and FreeWalk Mode.
- 6) When a book, shelf or librarian is chosen by the user, incoming states from the Graphic Component are stored in the state bank of the component.
- 7) Output of the component is manipulated by the changes on the states.

The detailed explanation of the methods used for mentioned operations is left for the Detailed Design section.

Class Diagram - World Model Component

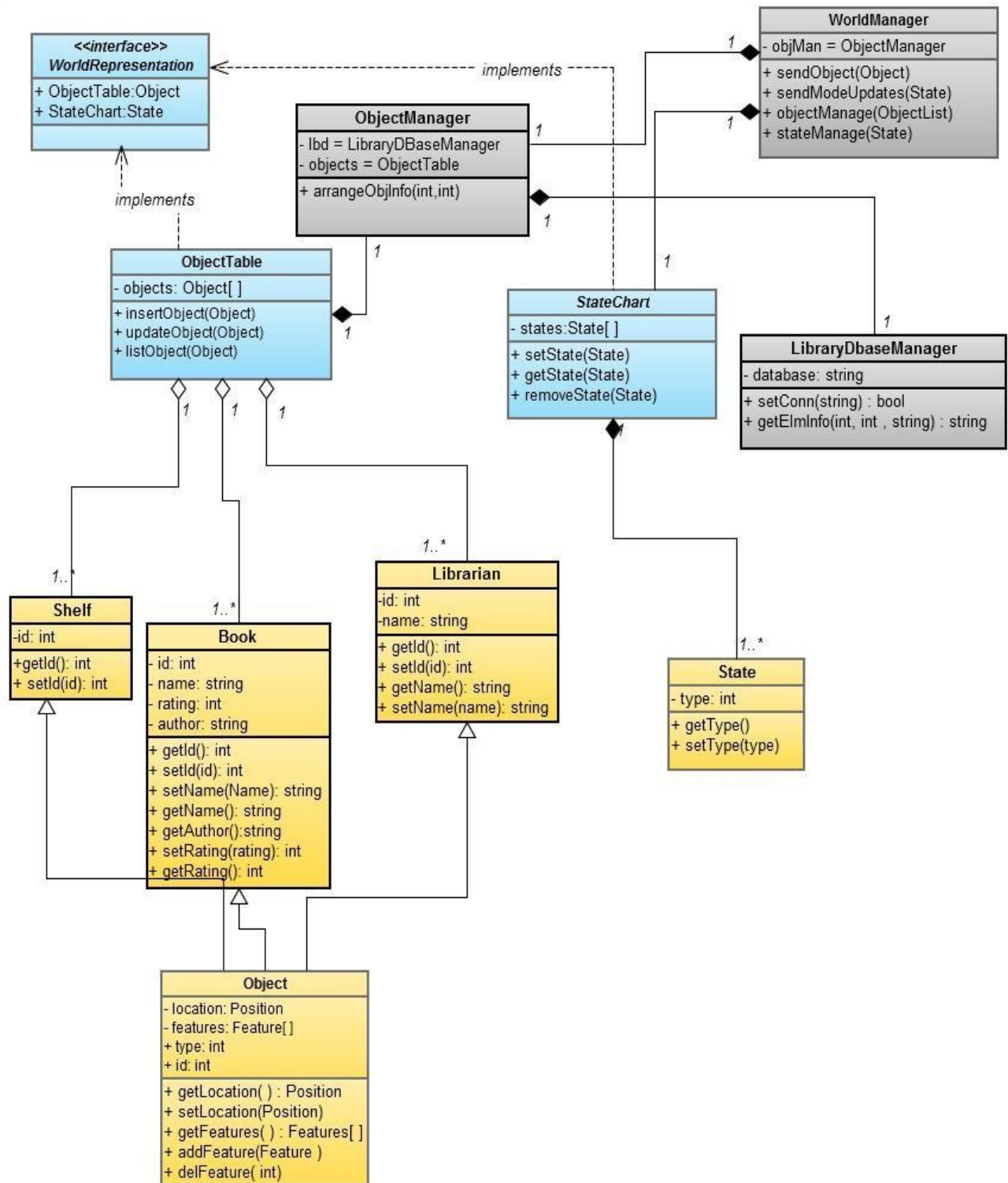


Diagram 5.7 – Class Diagram

5.2.2.4 Dynamic Behavior of World Model Component

In the World Model Unit, the interactions between the components is explained not in a very detailed manner, but in an initial detailed design manner.

After the information coming from the Object Recognizer Unit is processed firstly in the WorldManager class. WorldManager class interacts with two basic classes namely: ObjectManager and StateChart.

Object Management:

- WorldManager interacts with two basic classes, namely: ObjectManager and StateChart. These classes implements an interface called WorldRepresentation.
- ObjectManager interacts with the library database manager , LibraryDBaseManager and ObjectTable .
- The interaction between the ObjectManager and ObjectTable is processed after the interaction between ObjectManager and LibraryDBaseManager is established. This sequence is clearly shown in the diagram.
- ObjectManager 's interaction with the ObjectTable is established, the specific objects comes to the interaction. Since our objects are mainly books, shelves or librarians, Book, Shelf and Librarian classes are inherited from the Object class.
- ObjectTable gathers information from the Book, Shelf and Librarian instances by using their specific methods.

Sequence Diagram - Object Management

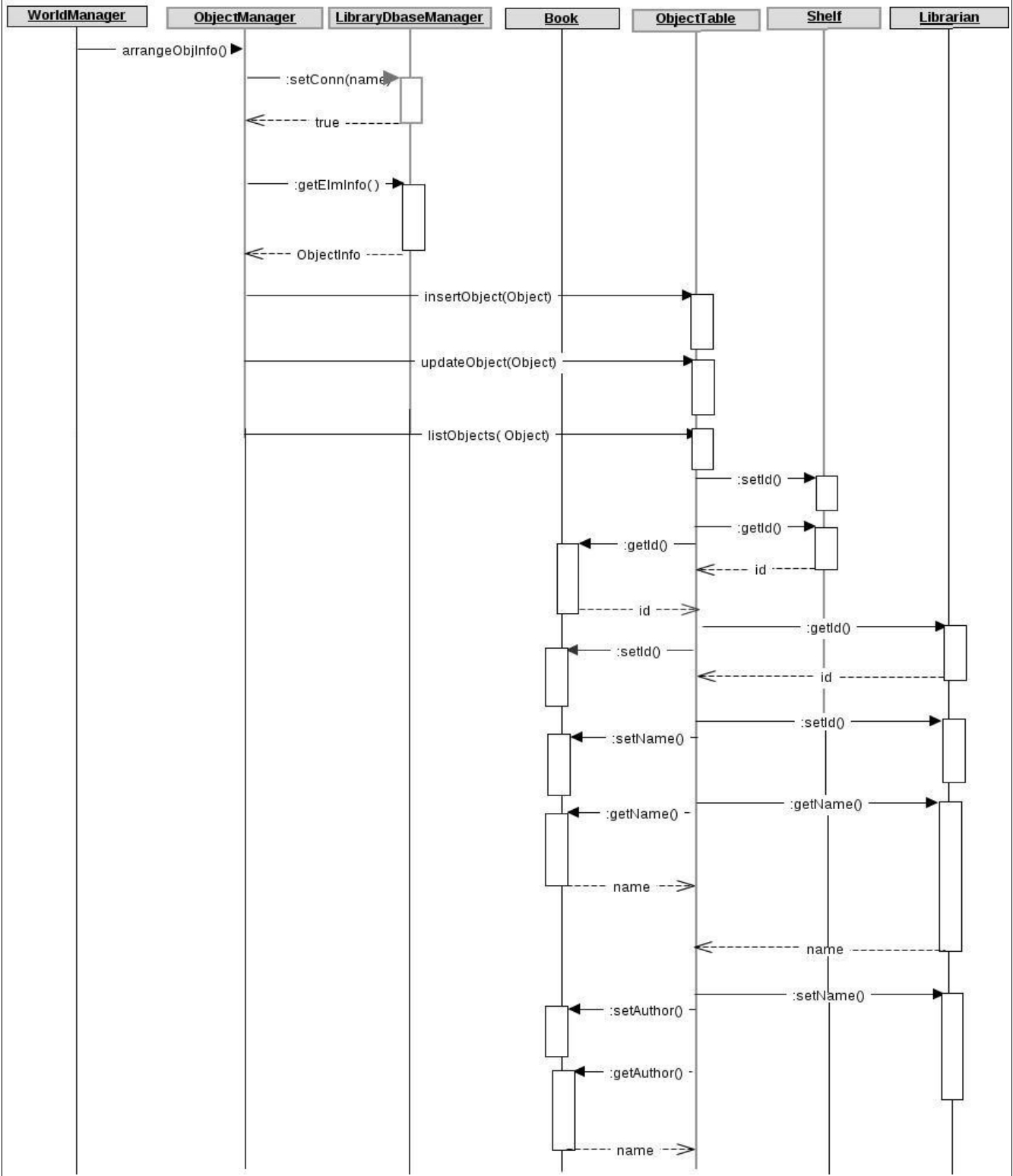


Diagram 5.8 – Sequence Diagram

State Management:

Now, it's time for the second class that the WorldManager interacts with, namely, StateChart. The following explanations and related sequence diagram as follows:

- WorldManager Unit interacts directly StateChart class.
- A State instance is created in order to answer the state changes in the Bookwiser.
- When a State is expired, the state is removed from the StateChart via `removeState()` call.

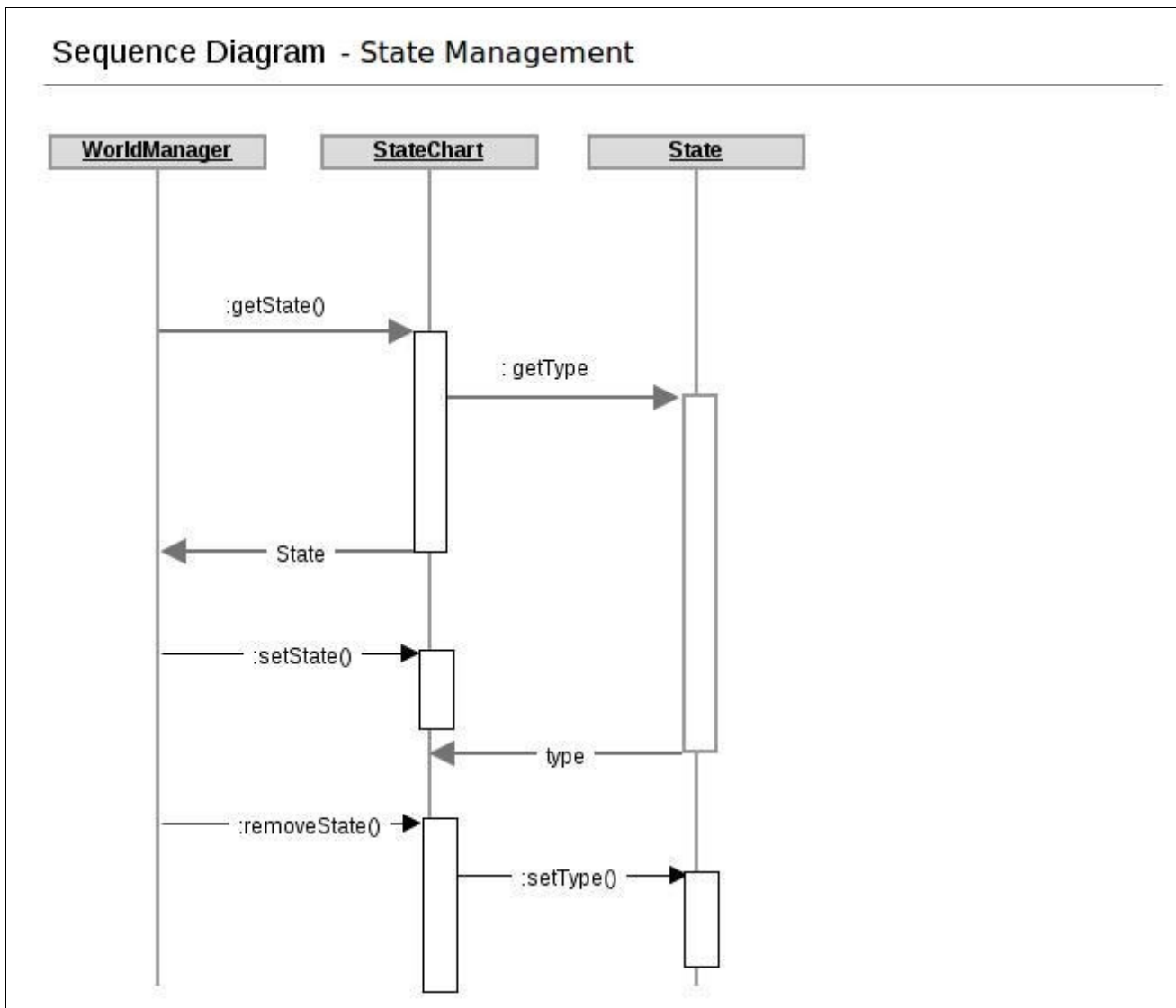


Diagram 5.9 – Sequence Diagram

5.2.3 Graphic Component

In this section , an explanation for the Graphic Component and the related class and sequence diagrams are provided.

5.2.3.1 Processing Narrative for Graphic Component

This is the third component of Bookwiser Software Project. Graphic Unit is the component that directly interacts with the user. This component does not process on anything, but just controls the button clicks,the user interface and the functionalities that directly lies between the user and the system without any calculation like updating the rate of some book or reserving some book.

Graphic Unit consists of three parts; video composer, input controller and graphical user interface, respectively.

5.2.3.2 Interface Description for Graphic Component

The interfaces that Graphic Component has is with the second main component of Bookwiser System, World Model Unit and with the Library Database.

- The Video Composer subcomponent of Graphic Component interacts with the World Model Component via WorldRepresentation interface as an input interface. This subcomponent combines the contextual and visual object information taken from World Model Component and the video stream captured by the camera. So, it must interact with the second main component.
- The second interface also is with second main component and is an output interface called States. Since handling the buttons is one of the missions of Graphic Component, also sending the states for different user choices to the World Model Component is its duty.
- The last interface that the Graphic Component deals with is with the library database. When there is a need of an direct on the update the Graphic Unit handles it. The updates can be reserving, unreserving or rating a book.

5.2.3.3 Graphic Unit Processing Details

Video Composer:

Video composer is the simplest section of the Graphic Unit component. The Video Composer unit combines the captured video stream by the camera with the object information coming from the World Model Unit. The combined video is sent to the GUI to be translated to the user.

Input Controller:

Input controller is based on two subunits: Screen Based Action Handler and Record Based Action Handler, namely.

Screen Based Action Handler:

Screen Based Action Handler subunit is responsible for the button-click handles and mode changes.

There are two button-click handles concerning the screen information: displaying detailed information for a selected shelf and displaying detailed information for a selected book. World Model Unit is informed about these button-clicks to process the objects as required. The mode changes also are handled in this subunit. Whenever the mode is changed (the mode can be free-walk mode or search mode) the World Model Unit is informed.

Record Based Action Handler:

Record Based Action Handler is related to the direct database updates, which there is two of them : reserving a book and rating a book.

The Record Based Action Handler subunit directly connects to the library database to update the rate or the reservation information of some book according to the information coming from user.

Graphical User Interface:

Graphical User Interface is the subcomponent of Graphical Unit that directly interacts with the user. It directly deals with the user's demands and process according to those.

Also, this subcomponent takes the composed video coming from the Video Composer Unit and displays that to the user.

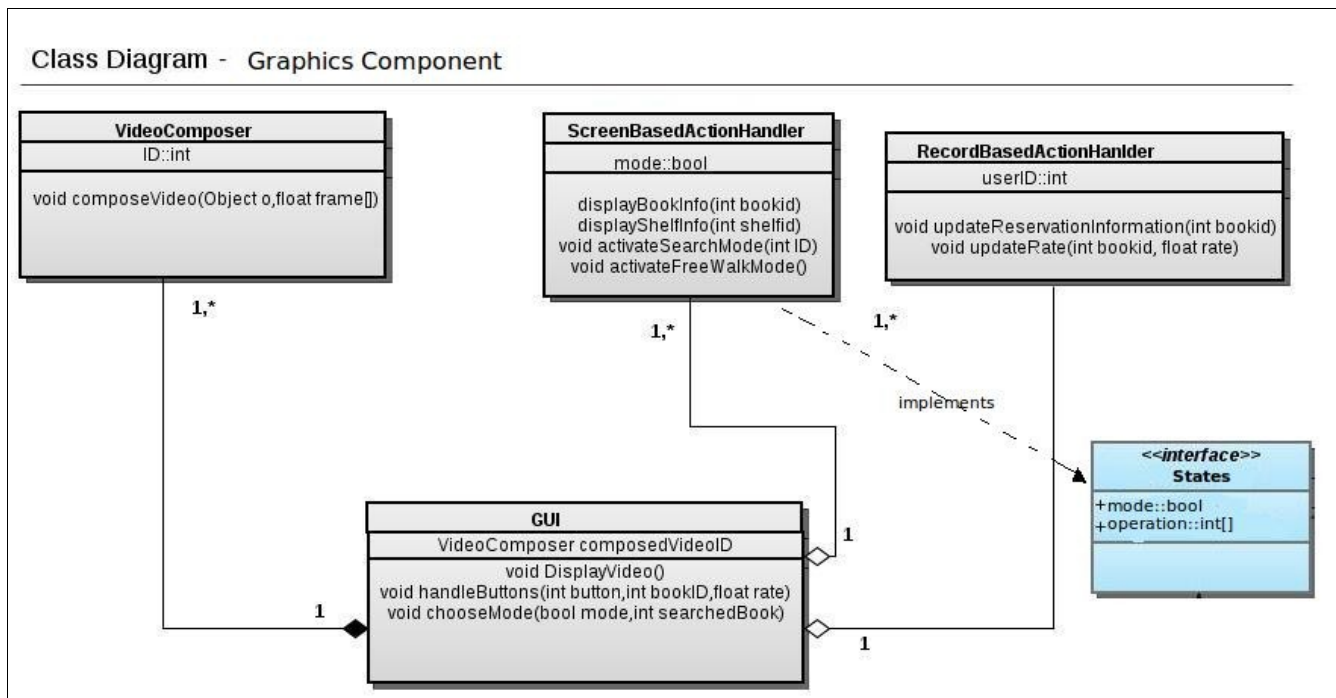


Diagram 5.10 – Class Diagram

5.2.3.4 Dynamic Behavior of Graphic Component

Choosing Objects:

- `handleButtons(1,shelfID)` method of GUI instance is calling `displayShelfInfo(shelfID)` method of ScreenBasedActionHandler instance which is going to request the information about the given shelf.
- `handleButtons(2,bookID)` method of GUI instance is calling `displayBookInfo(bookID)` method of ScreenBasedActionHandler instance which is going to request the information about the given shelf.

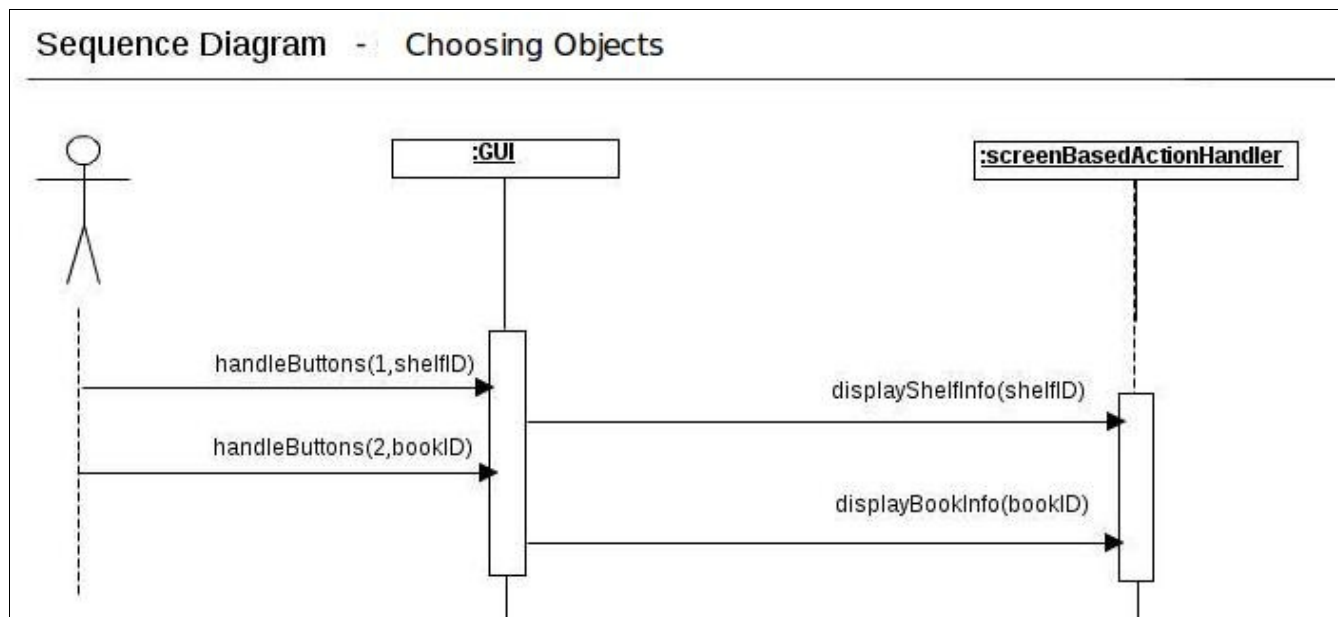


Diagram 5.11 – Sequence Diagram

Choosing Modes:

- `chooseMode(true,ID)` method of GUI instance is calling `activateSearchMode (ID)` method of ScreenBasedActionHandler instance which is going to request an alert when the location of the book with the given ID is in the vision of the user. The system mode is in search mode.
- `chooseMode(false,ID)` method of GUI instance is calling `activateFreeWalkMode ()` method of ScreenBasedActionHandler instance which is going to change the mode of the system to free-walk mode. In this situation the argument 'ID' is not used, is ignored.

Sequence Diagram - Choosing Modes

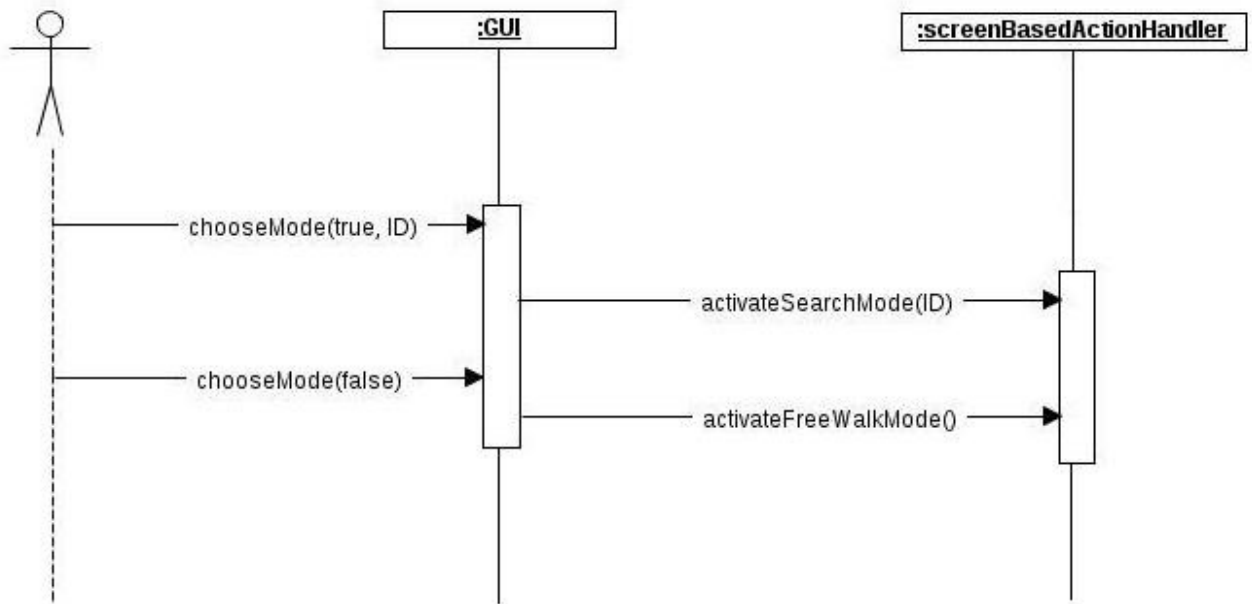


Diagram 5.12 – Sequence Diagram

Record Updates:

- handleButtons(3,bookID) method of GUI instance is calling updateReservationInformation (bookID) method of RecordBasedActionHandler which is going to update the reservation possession of the book with the given bookID.
- handleButtons(3,bookID,9,5) method of GUI instance is calling updateRate (bookID,9,5) method of RecordBasedActionHandler which is going to update the rate of the book with the given bookID and with the given rate, 9.5 in this sequence diagram.

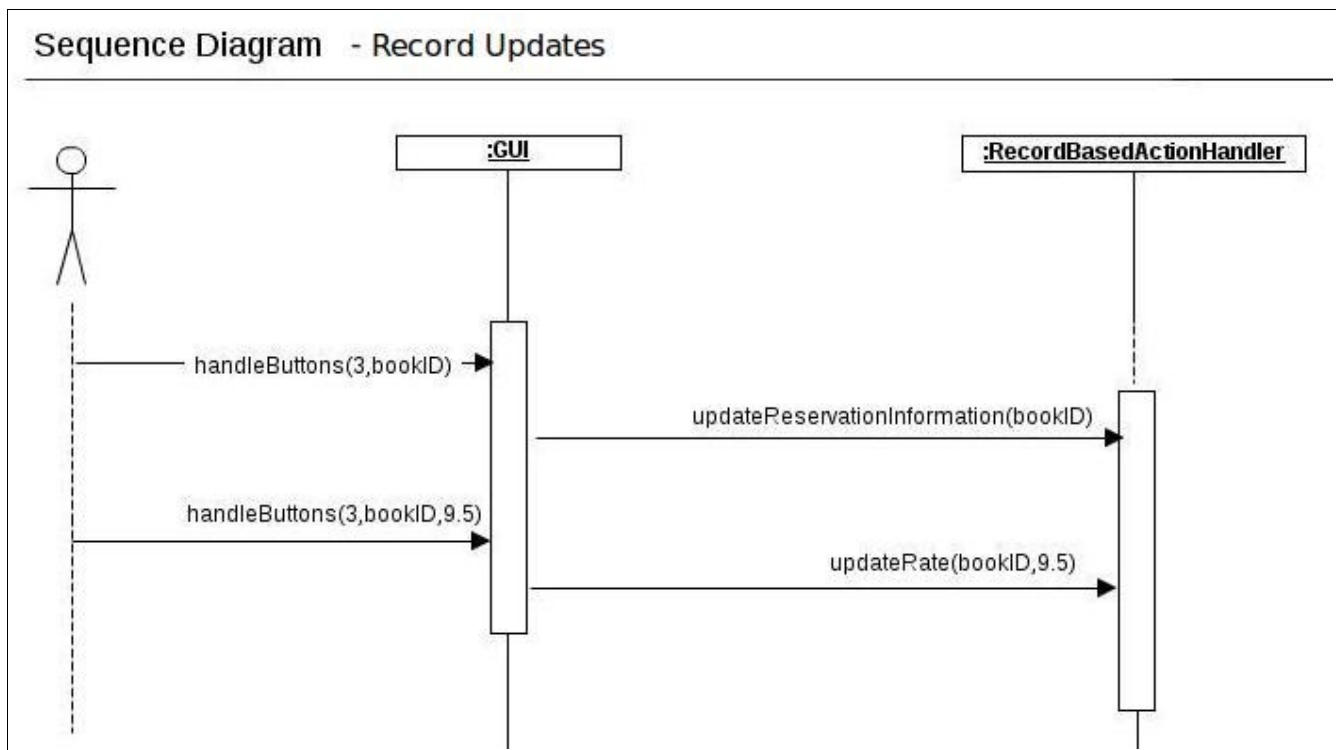


Diagram 5.13 – Sequence Diagram

5.3 Design Rationale

The abstraction of three components is directly arisen from the need to handle three external elements which system interacts: camera, user and library database. Object recognizer and graphic unit are exactly designed for handling two of them.

World model unit on the other hand is not designed for handling library database. Although it use library database to get contextual information, this is only a realization of its function. The idea that creating a world model for library reduces complexity of the system greatly and provides a basis for all sub-components of the system when they perform their tasks according to world state. Upon this architecture, it is possible to improve Bookwiser's functionality and enhancing its capability with extra services.

6 User Interface Design

In this section user interface design details are provided.

6.1 Overview of User Interface

Bookwiser's user interface is the most critical unit from user's perspective. All the interaction between Bookwiser and user is done through user interface. Main job of user interface is showing the user the video stream enhanced with extra information. It is also responsible for capturing user actions such as mode selection, book selection etc.

6.2 Screen Images

Screen shots of prototype user interface are given in this section:

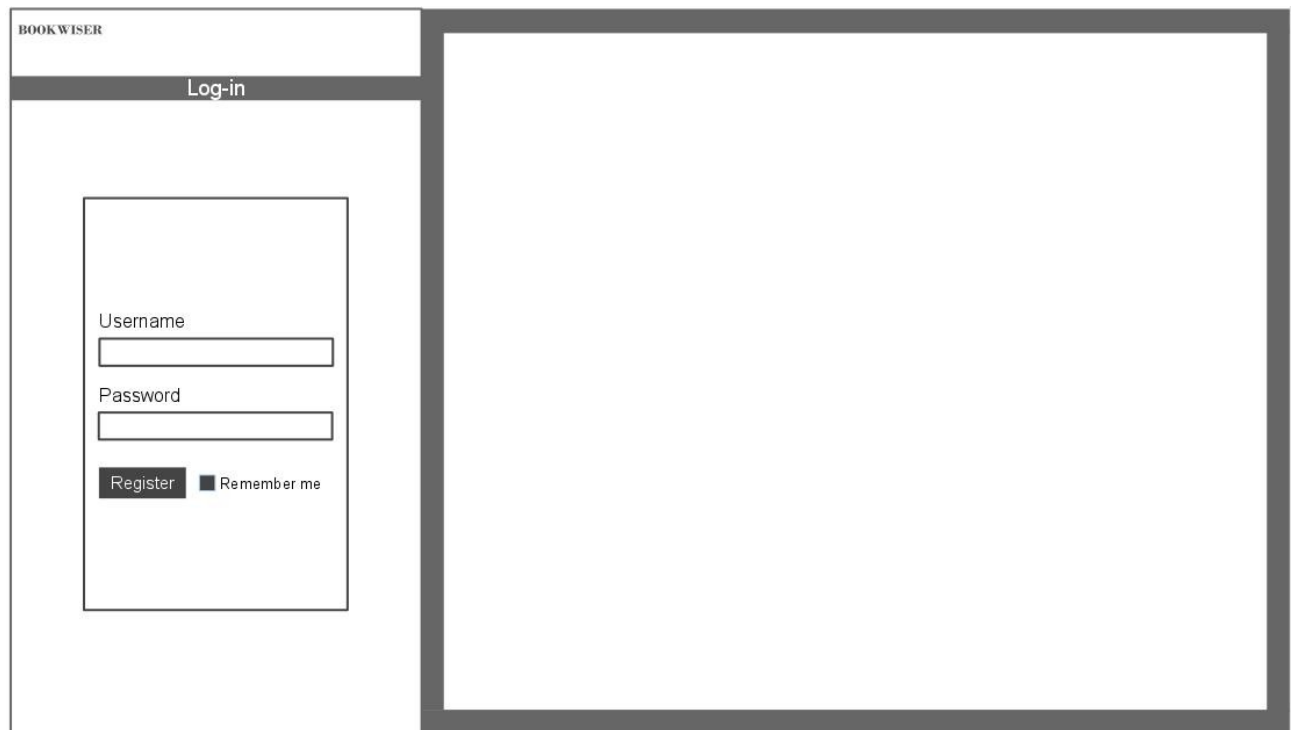
Register:

The screenshot shows a registration form within a browser window titled "BOOKWISER". The form has a sub-header "Registration". Below this, it says "Please enter following information:". The form contains the following fields:

- Name
- Surname
- Date of Birth
- E-mail
- Nationality
- Address
- Phone Number
- Preferred Language
- Password
- Re-type Password

At the bottom of the form is a button labeled "FINISH".

Log-in:



BOOKWISER

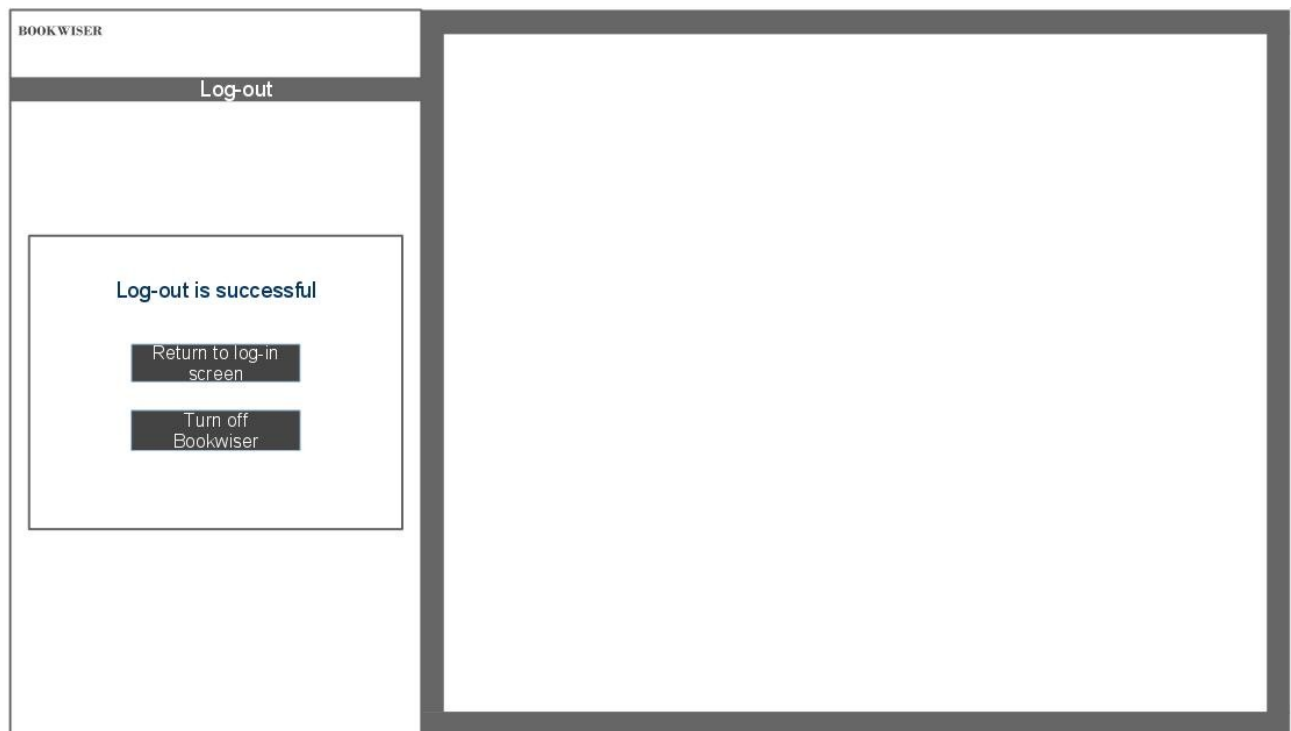
Log-in

Username

Password

Remember me

Log-out:

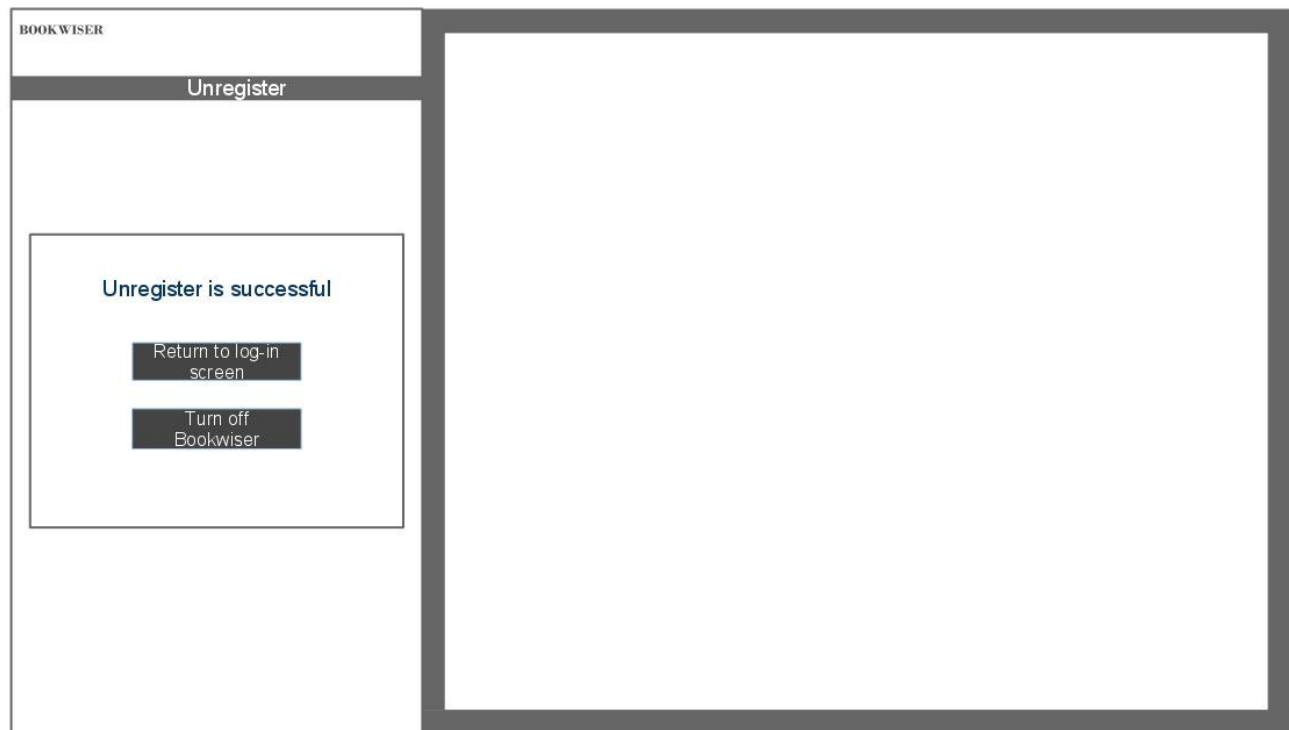


BOOKWISER

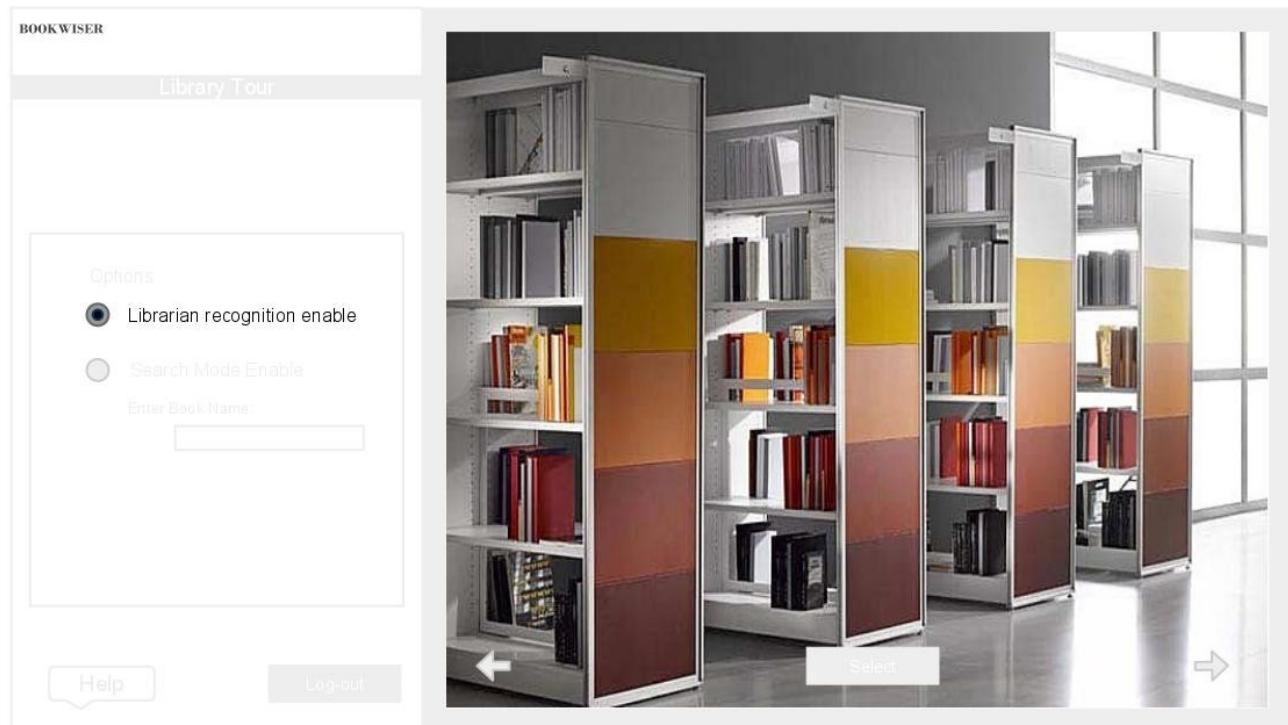
Log-out

Log-out is successful

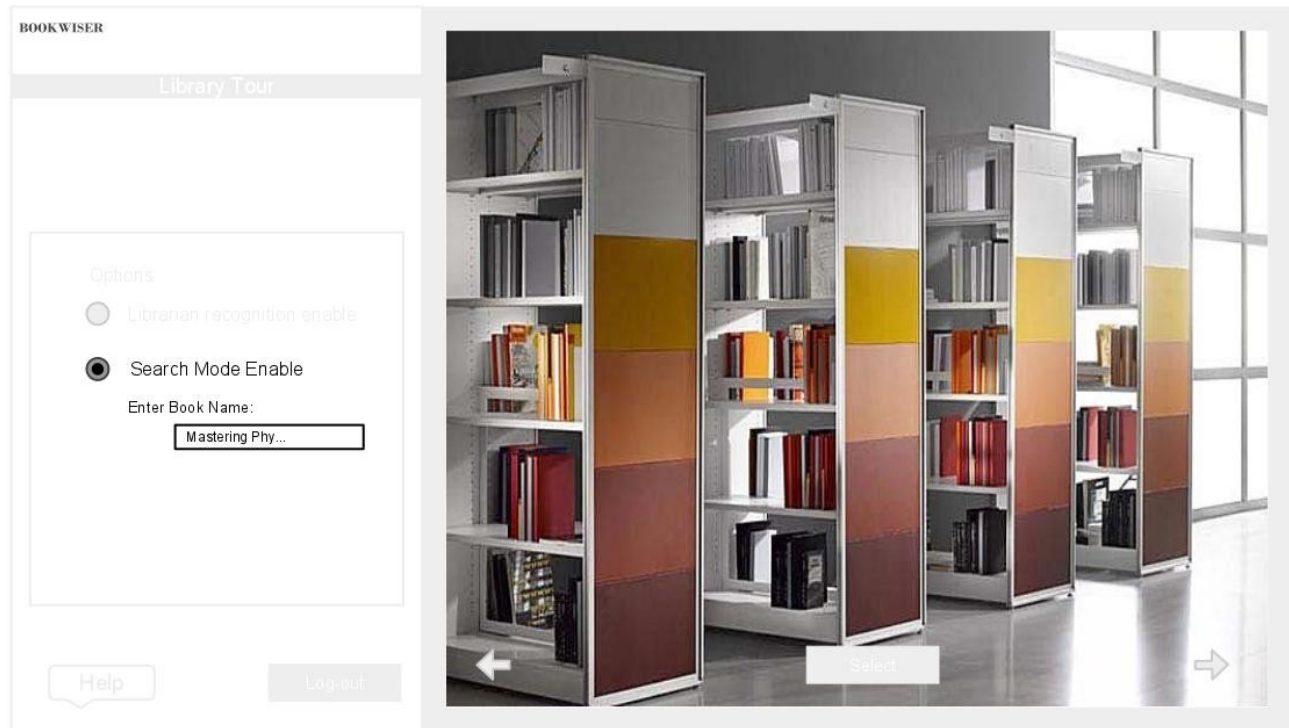
Unregister:



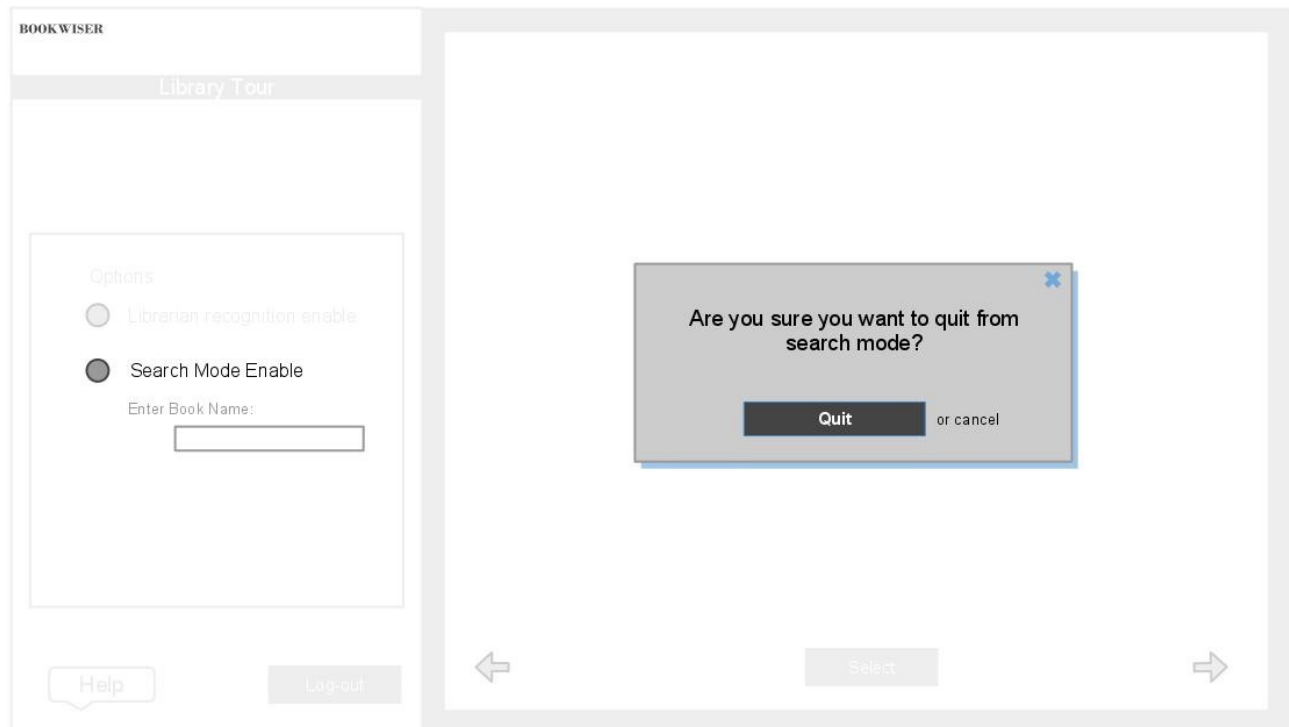
Librarian Recognition:



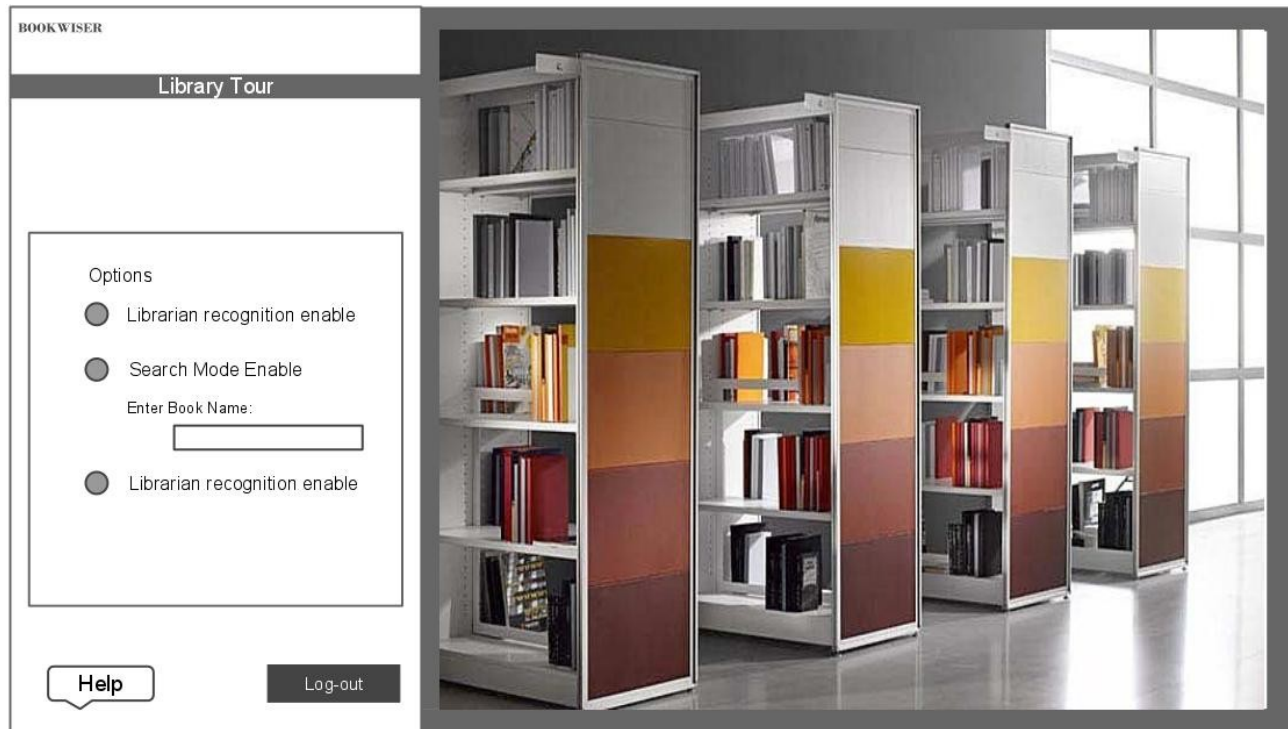
Search Mode:



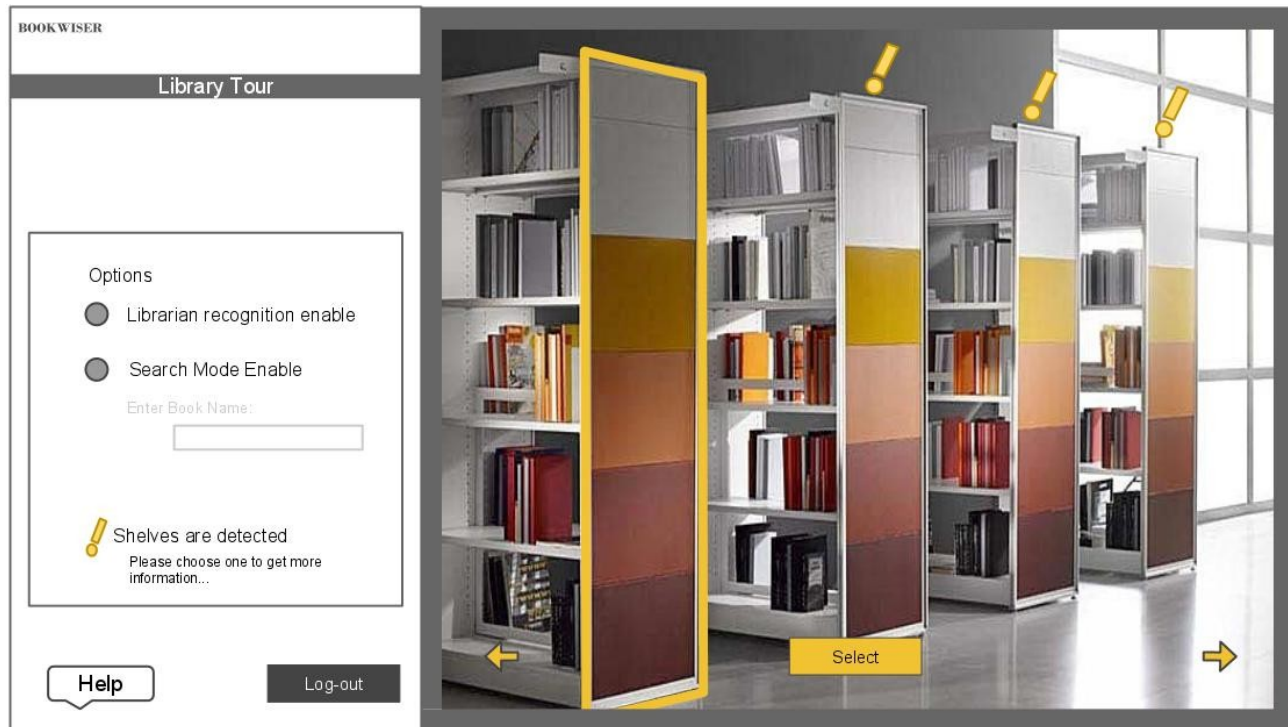
Cancel search mode:



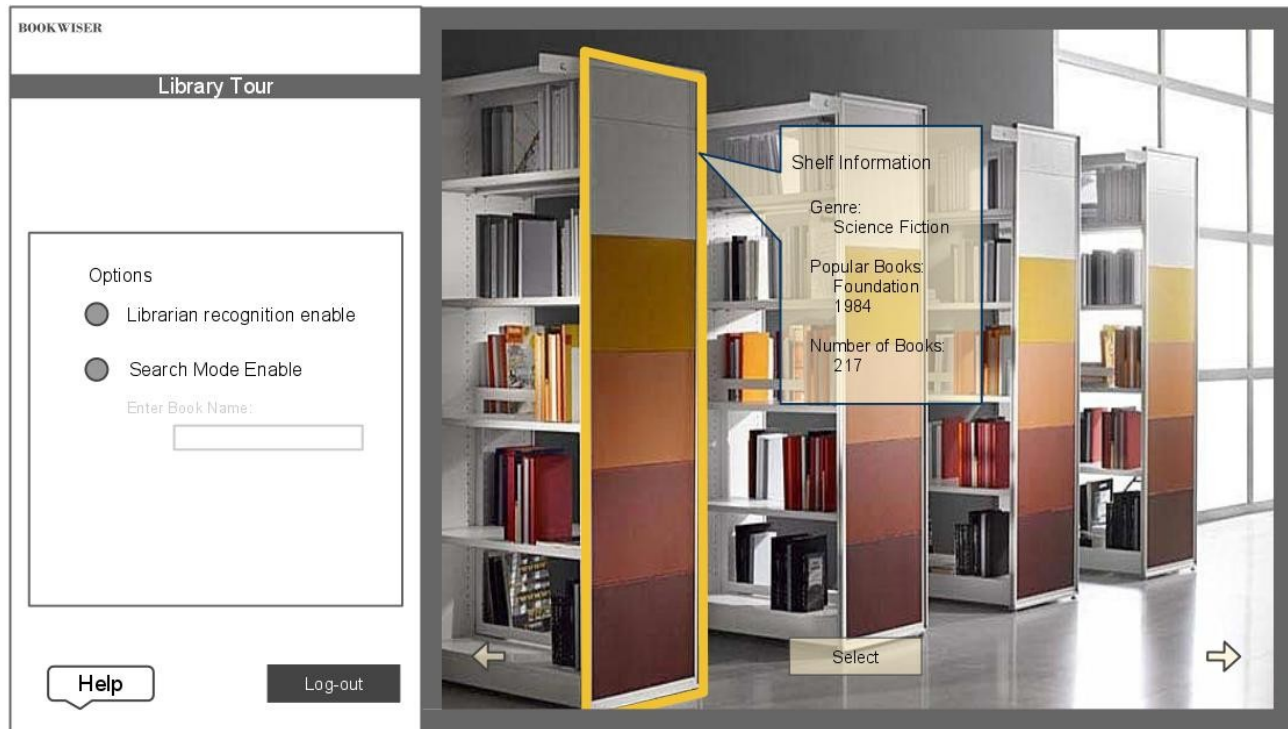
Freewalk Mode:



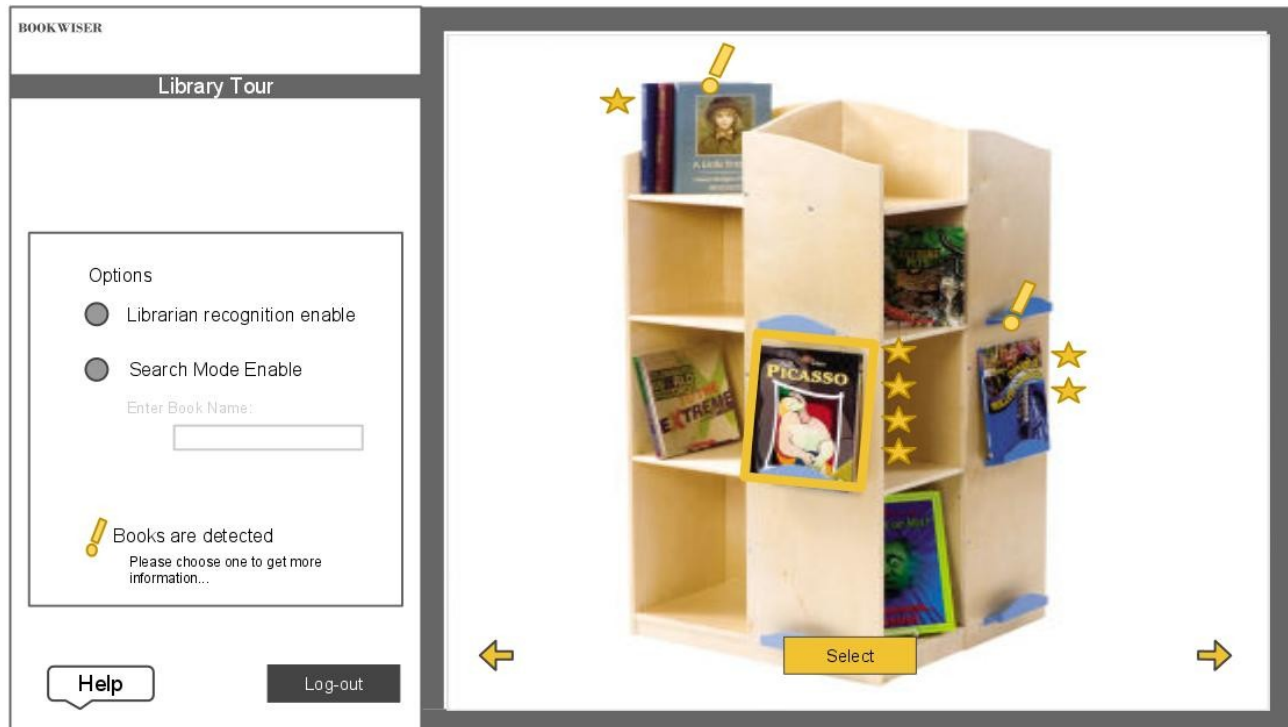
Recognizing shelves:



Request shelf info:



Recognizing books:



Selecting book:

BOOKWISER

Book Information

★★★★★

PICASSO
by Batuhan Karagöz

Ballmer Peak Yayınevi

1967

Golden Edition


Abstract
This book of Karagoz is ...

☆ Rate the book

📄 Get detailed information

📄 Reserve the book

Help Log-out



Rating book:

BOOKWISER

Book Information

★★★★★

PICASSO
by Batuhan Karagöz

Ballmer Peak Yayınevi

1967

Golden Edition


Abstract
This book of Karagoz is ...

☆ Rate the book

📄 Get detailed information

📄 Reserve the book

Help Log-out



Request detailed book info:

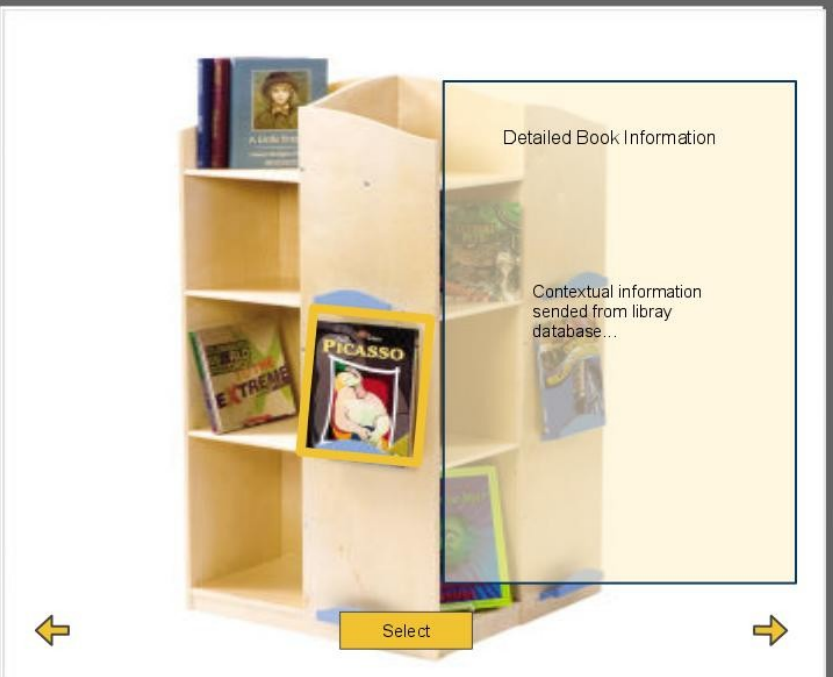
BOOKWISER

Book Information

★★★★★
PICASSO
by Batuhan Karagöz
Ballmer Peak Yayınevi
1967
Golden Edition
Abstract
This book of Karagoz is ...

☆ Rate the book
! Get detailed information
📄 Reserve the book

Help Log-out



Detailed Book Information

Contextual information
sent from library
database...

List books of the author:

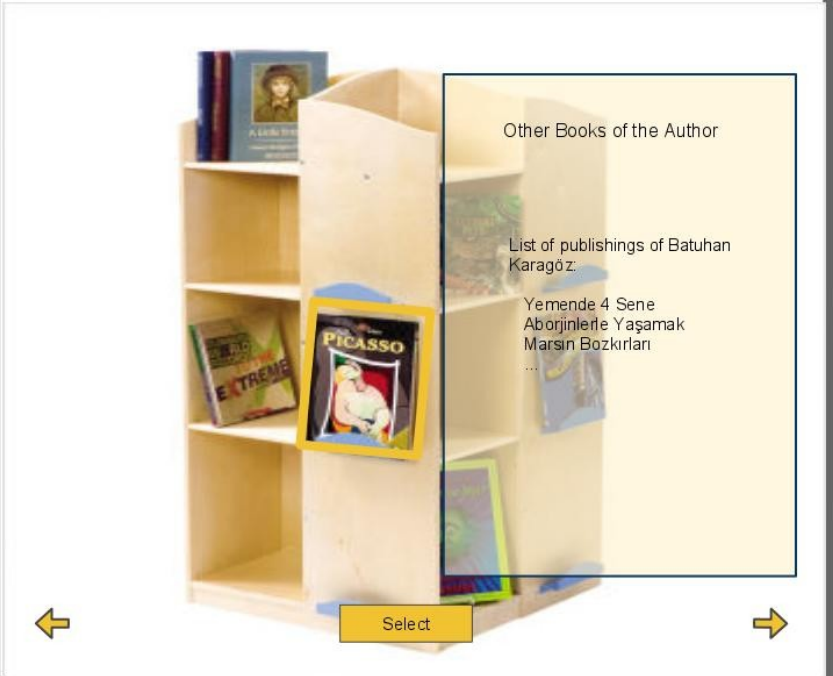
BOOKWISER

Book Information

★★★★★
PICASSO
by Batuhan Karagöz
Ballmer Peak Yayınevi
1967
Golden Edition
Abstract
This book of Karagoz is ...

☆ Rate the book
! Get detailed information
📄 Reserve the book

Help Log-out



Other Books of the Author

List of publications of Batuhan
Karagöz:
Yemende 4 Sene
Aborjinlerle Yaşamak
Marsin Bozkırları
....

Reservation:

BOOKWISER

Help

★★★★★

PICASSO
by Batuhan Karagöz

Reserve

Current Reservations
Şafağa Geçit


Extend Cancel

Yitik Güneş

Extend Cancel

Back to the info screen

Help Log-out




Help:

BOOKWISER

Help

- General information
- Modes
 - Search mode
 - Free walk mode
- Display Features
 - Shelves
 - Shelf recognition
 - Shelf genres
 - Selecting a shelf
 - Books
 - Book recognition
 - Book ratings
 - How to rate?
 - Selecting a book
 - Request more info
 - Detailed info
 - Books of the author
 - Reservation
- Rules
- Options

Back Log-out



6.3 Screen Objects and Actions

There are two parts of a Bookwiser screen. The left side of the screen is occupied by the system menu, while video is shown at the right side.

After login, there is always a log-out button in the menu display. Menu display has several screens according to user actions. User can able to see all information about the Bookwiser state, set options and do particular tasks about the object records such as requesting extra information, reservation etc.

On the left hand side there is camera view with some extra elements. Most basically, when encountered with several objects ,books or shelves, two arrows and one select button are displayed. These are selection buttons for book selection and shelf selection. Multiple objects can be examined by using this buttons. Along with these selection buttons, there were also specific display elements such as stars and info tabs. These display elements allows user to retrieve information easily from camera view.

7 Libraries and Tools

In this section, used libraries and tools are described.

1. OpenCV:

In the Bookwiser project, OpenCV library is used for the object detection part. OpenCV is an open source computer vision library in C/C++. Since it is optimized and intended for real-time applications, it is very applicable to use in Bookwiser which is a real-time augmented reality application. Bookwiser is designed to be platform independent in a way that it will work on the mobile devices. Independence of operating system/hardware or window-manager, OpenCV is the best choice for our purpose. Moreover, it provides a generic image/video loading and both low and high level of API.

OpenCV has considerably high quality features that provides so much easiness to the developers:

1. Image and video I/O (file and camera based input, image/video file output)
2. Various dynamic data structures (lists, queues, sets, trees, graphs)
3. Basic image and video manipulation processing such as filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids.
4. Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation)
5. Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence)
6. Motion analysis (optical flow, motion segmentation, tracking)
7. Object recognition (eigen-methods, HMM).

For all the reasons listed above, **Bookwiser** uses the OpenCV.

2. QT Tool:

For the graphical user interface of Bookwiser, QT tool is used.

QT brings many advantages as portability, predefined interface with OpenCV and useful built-in tools for GUI development.

8 Gantt Chart

| ID | Task Name | Duration | Start | Finish |
|----|------------------------------|----------|--------------|--------------|
| 1 | Literature Survey | 8 days? | Mon 11.10.10 | Wed 20.10.10 |
| 2 | General Design | 11 days? | Fri 15.10.10 | Fri 29.10.10 |
| 3 | Database Design | 5 days? | Mon 01.11.10 | Fri 05.11.10 |
| 4 | Library Database Entegration | 4 days? | Mon 08.11.10 | Thu 11.11.10 |
| 5 | Basic User Interface Impleme | 4 days? | Fri 05.11.10 | Wed 10.11.10 |
| 6 | Authentication | 3 days? | Thu 11.11.10 | Mon 15.11.10 |
| 7 | Shelf Recognition | 5 days? | Tue 16.11.10 | Mon 22.11.10 |
| 8 | Book Recognition | 10 days? | Tue 16.11.10 | Mon 29.11.10 |
| 9 | Librarian Recognition | 6 days? | Mon 22.11.10 | Mon 29.11.10 |
| 10 | Library Network Interface De | 5 days? | Tue 30.11.10 | Mon 06.12.10 |
| 11 | User Interface Development | 8 days? | Wed 01.12.10 | Fri 10.12.10 |
| 12 | Team Presentation | 1 day | Mon 13.12.10 | Mon 13.12.10 |
| 13 | Demo Release | 3 days? | Tue 14.12.10 | Thu 16.12.10 |
| 14 | Demo Test | 4 days? | Fri 17.12.10 | Wed 22.12.10 |
| 15 | Project Improvement | 7 days | Thu 23.12.10 | Fri 31.12.10 |
| 16 | Product Release | 5 days? | Mon 03.01.11 | Fri 07.01.11 |
| 17 | Product Test | 4 days? | Mon 10.01.11 | Thu 13.01.11 |



9 Conclusion

Bookwiser is a valuable project that will both serve the users and the owners of the system in a positive way, being a innovative idea and aiming to create a more comfortable environment for its users.

The project should be developed in a complete understanding of the facts behind the need for the project, the importance of the results of the project and should be considered as a part of the instinct that creates the need of progress.

Initial design is the first design of the project in a way that further detailed information comes up with this document. This document gives design facts such that how the system, Bookwiser, architecture is built and how the information flows in the components of projects. Moreover, it defines all the components with an initial details by class diagrams and sequence diagrams, user interface design and how the user interacts with Bookwiser, structures, limitations and planning to guide the developer through all the development process.

All the initial details stated in this document is the result of a mixture containing innovative thinking, analytic approach and a proper analysis of users' needs; therefore, developers should not forget that these details have a critical importance for all elements of all levels in the project as a whole.

All the initial details stated in this document should give the developers a design mainframe with some details.