**CHAPTER 3**

**SYSTEM SPECIFICATION**

System specification is a structured collection of information that embodies the requirements of the system. This section provides a overview of the entire system or product. This chapter describe the subsystem including hardware requirements, software requirements of the Asset Management System.

**HARDWARE SPECIFICATION**

**SYSTEM :**

**HARD DISK :**

**FLOPPY DRIVE :**

**SOFTWARE SPECIFICATION:**

**FRONTEND :** CUBA

**BACKEND :** HSQL

**OPERATING SYSTEM :** Win 7

**LANGUAGE :** JAVA

**CHAPTER 4**

**SOFTWARE DESCRIPTION**

This chapter gives the description of the software used and their important features. Software used to create Asset Management System has certain features that help enhance this project and improve its scalability.

**3.1 Front End**

The front end is designed using Cuba Platform and it is a high level open source Java Framework and the intermediate language used is Java.

**Frond End &Other Technologies**

CUBA Platform is a high level open source Java web framework for the rapid development of enterprise applications with rich web interfaces. The platform abstracts developers from underlying [technologies](https://www.cuba-platform.com/framework#technologies) so they can focus on the business tasks, whilst retaining full flexibility by providing unrestricted access to low level code.

Applications are developed in Java, with the user interface declared in XML. There

is no need for HTML, CSS or JavaScript to create standard web interfaces. A rich set of [features](https://www.cuba-platform.com/framework#features) covers most typical project requirements and [development tools](https://www.cuba-platform.com/development-tools) reduce boilerplate code and facilitate truly rapid development.

The unique Extensions concept enables developers to customize a distributable product to individual client requirements whilst retaining the benefits of all platform features and future updates - delivering bespoke products without branching.

CUBA Platform is available under the open source Apache 2.0 license, which ensures the freedom to use the platform and your applications for any purpose.

**Problem definition:**

The Assets Management System helps in creating and managing the assets effectively using the auditing network. It keeps track of a number of assets that can be borrowed, their ownership, their availability, their current location, the current borrower and the asset history. Assets include software, computers, and peripherals. Assets are entered in the database when acquired, deleted from the database when disposed. The availability is updated whenever it is borrowed or returned. When a borrower fails to return an asset on time, the asset management system sends a reminder to the borrower and informs the asset owner.

The administrator enters new assets in the database, deletes obsolete ones, and updates any information related to assets. The borrower search for assets in the database to determine their availability and borrows and returns assets. The asset owner loans assets to borrowers. Each system has exactly one administrator, one or more asset owners, and one or more borrowers. When referring to any of the above actor, we use the term "user". All users are known to the system by their name and their email address. The system may keep track of other attributes such as the owner's telephone number, title, address, and position in the organization.

The administrator can create one or more assets using a web form. Each asset includes a name, a unique identifier, a brief description, an owner and a status. The administrator can also delete one or more specified assets. The assets can be deleted from the system only if they are available. The administrator can also modify any information related with an asset, except for its status which is modified only by the asset owner.

A user can search the asset information using any asset attribute or using free text search. The system displays references to all matching assets, sorted by a user specified criteria. Once an asset record is displayed to the user, the user browse related assets. A user can browse the asset information by listing all references to assets sorted by a user specified criteria.

After establishing that an asset is available, a borrower checks out an asset by visiting the owner of the asset. The owner uses a web form to record the borrower's name, email address, and other contact information. The system uses the email address to identify the borrower, and update any previous information on the borrower. The owner also enters a due date associated with the loan.

Once the asset is checked out and associated with the borrower, the system records the loan and updates the status of the asset. If the due date arrives without the asset being returned, the system sends an email message to the borrower listing all the assets that are past due. The owner of a past due asset also receives an email informing him/her of the delay.

The borrower returns the asset to the owner. The owner then searches for the asset record and marks the asset returned. If the status of the asset has changed, the owner also updates this information. The system takes out the loan from the list of active loans and archives it.

The system should support at least 200 borrowers and 2000 assets. The system should be extensible to other types of assets. The system should checkpoint the state of the database every day such that it can be recovered in case of data loss. Owners and the administrator are authenticated using a user/password combination. Actors interact with the system via a web browser capable of rendering HTML and HTTP without support for JavaScript and Java.

The persistent storage is realized using an SQL database. The business logic is realized using the WebObjects runtime system.

**Overview of the project:**