Software Requirements Specification

for



**Version 1.0**

Prepared By:

Tack Tack Poh (A0098802X)

Ho Tuck Wai James (A0081093J)

Lim Wancai, Daryl (A0096586J)

Doan Xuan Loc Nguyen (A0099858Y)

Qiu Yunhan (A0101735R)

Submitted in fulfilment of requirements for CS3283 / CS3284

National University of Singapore

Table of Contents

[1. Introduction](#h.pl6gu8pq8qbg)

[1.1 Purpose](#h.1hf2hbqaxkfg)

[1.2 Product Scope](#h.zbzn9pk9if5e)

[2. Overall Description](#h.796v43d9uxa7)

[2.1 Product Perspective](#h.x0k2dse39vk0)

[2.2 Product Functions](#h.kde16n5cxjnk)

[2.3 User Classes and Characteristics](#h.bwwuqof4qn2d)

[2.3.1 Corporate/Organisation](#h.ahpndkxcdfgo)

[2.3.2 Professionals / High-Med Income Users](#h.w3yc6lb43ma0)

[2.3.2 Households](#h.4wkfwhdywa8p)

[2.4 Operating Environment](#h.shazb5h2uwnd)

[2.5 Design and Implementation Constraints](#h.6ymhfaqhfkn9)

[2.5.1 OCR Accuracy](#h.tk70mkmvoev6)

[2.5.2 Server Space/Bandwidth/Processing Limitation](#h.jhnb5py38cbs)

[2.5.3 Digitization Quality](#h.vqrxtoz5yuyw)

[2.5.4 User Platforms](#h.amo9yjqyqaff)

[2.5.5 Legal Constraints](#h.alumc918akkl)

[2.5.6 Connectivity/Hardware Issues](#h.z3z6t0qpr4xv)

[2.6 User Documentation](#h.xr6wfd7d79z3)

[2.7 Assumptions and Dependencies](#h.6mod4ng0m0of)

[2.7.1 Third Party Libraries](#h.8fizexxpk1xp)

[3. External Interface Requirements](#h.37dvkevfk7ef)

[3.1 User Interfaces](#h.p1x45uga2iz4)

[3.1.1 Design Conventions](#h.7alk2frntbbr)

[3.1.2 UI Interactions](#h.vpkh2dpqscw8)

[3.1.2.1 Main Application UI](#h.benha7q8n00f)

[3.1.2.2 Bill Upload](#h.m5ge6tynwr9q)

[3.1.2.3 Image Editing UI](#h.pugyeanu9ned)

[3.1.2.4 Template Manager UI](#h.81ujwwkm7cr)

[3.2 Software Interfaces](#h.up6tviw31rlh)

[3.3 Communications Interfaces](#h.lr7ksr9jmbmz)

[4. System Features](#h.luibreq3sy4p)

[4.1 Data Capture/Upload](#h.nkm5vdoepox0)

[4.2 User Account Management](#h.2lwtdjv9klvq)

[4.3 Pre-Processing Bills](#h.uzyyaaydve0x)

[4.4 Tagging and Sharing](#h.kk1mlcix1qmg)

[4.5 Processing (Recognition) & Templating](#h.6e0ph61zie24)

[4.6 Data Analysis / Dashboard](#h.dphr69mto11n)

[4.7 Bill Retrieval & Management](#h.m7g8wnbrdwj2)

[5. Other Nonfunctional Requirements](#h.m9jm91ih1cs6)

[5.1 Performance Requirements](#h.sq3pqz993xrc)

[5.1.1 Client-side Performance](#h.3j4pq2fmm2xc)

[5.1.2 Server-side Performance](#h.5lysyilqzkhw)

[5.2 Security Requirements](#h.ale5sxj7f3p5)

[5.3 Software Quality Attributes](#h.mpopag3y3586)

[5.3.1 For Developers](#h.94uvf2uj5nfo)

[5.3.2 For Users](#h.t8fplxidgyju)

[5.4 Operating Principles](#h.jq3mh88d8thr)

[Appendix A: Glossary](#h.3wsc5cs2rk8a)

# 

# 

# 1. Introduction

## 1.1 Purpose

This document aims to present a detailed description of the web-based bill/expense management and organisation system, henceforth referred to as “Bill.eGoat”. This document describes the purpose, features, interfaces, workflow, constraints and other functional or non-functional requirements. This document is intended for developers of Bill.eGoat and will be proposed to the client user, Prof. Ooi Wei Tsang, for approval.

## 1.2 Product Scope

Bill.eGoat is a web-based bill/expense management system mainly for users who receive multiple bills/statements from different organisations. These include recurring/ad-hoc bills such as phone/utilities/credit card/bank/medical/insurance statements and bills, henceforth simply referred to as “bills”.

Bill.eGoat aims to maximize the user’s efficiency in sorting and tracking such bills, while minimizing errors that may potentially result in financial penalties (e.g, late payment fees). Bill.eGoat does this by semi-automating most of the bill capture, tracking, analysis and organization process while still providing the tools for manual entry and review.

Specifically, users upload digital copies of bills to Bill.eGoat, which automatically sorts them according to their template and content. For instance, by billing organization or by date. Bill.eGoat can then analyse data from the bills and provide visual summaries or track selected information across billing dates. A template management interface is available for bills that are not within a list of pre-formatted templates.

# 2. Overall Description

## 2.1 Product Perspective

Bill.eGoat is a new, self-contained product that aims to combine the best features of several existing bill-management applications and Optical Character Recognition (OCR) / Pattern Recognition technologies. Current products for bill management (e.g, Shoeboxed) do not perform automatic sorting based on OCR/Pattern Recognition, nor do current products that use OCR provide integrated data analysis / visualisation (e.g, Evernote). Hence, Bill.eGoat is in a unique position to combine these functionalities for ease of use.

## 2.2 Product Functions

The following is a summary of the major functions that Bill.eGoat will allow users to perform, grouped by related functions. For further detail, see Section 4 - System Features.

1. Data Capture/Upload - This group of functions allow users to input/upload billing data into Bill.eGoat via manual data entry, uploading/emailing digitized bills / soft-copies.
2. User Account Management - This group of functions allow users manage their account with Bill.eGoat, including typical account authentication and login management, notification settings, financial account setup, known associates management and permission sharing.
3. Pre-Processing Bills - This group of functions allow users to pre-process their bill images before uploading, including allowing automatic rotation, resize, downsample, compress, crop, deskew and contrasting of digitized bills. Users will be able to manually adjust or perform pre-processing as well.
4. Tagging and sharing - This group of functions allow users to tag, categorise and define privacy/sharing options.
5. Processing (Recognition) & Templating - This group of functions allow users to define bill templates and edit/tag automatically recognised data.
6. Data Analysis / Dashboard - This group of functions allow users to analyse or summarise their billing data over a time period, as well as get a sense of their current financial position through a dashboard interface.
7. Bill Retrieval & Management - This group of functions allow users to edit individual bills’ data, permissions and flags, as well as search, revision control, annotation/redaction and notification tools.

## 2.3 User Classes and Characteristics

We anticipate various user classes which will make use of Bill.eGoat, and have included the more pertinent ones in this document for consideration.

### 2.3.1 Corporate/Organisation

Corporate users, or those belonging to significantly large organisations, would likely benefit most from Bill.eGoat’s automated sorting and management system, eliminating the need to carefully sort and analyse large numbers of bills. Corporations / Organisations are expected to have larger numbers of trained staff to handle actual payments.

**Proficiency/Technical Characteristics**

Such users would likely have high technical proficiency in using Bill.eGoat due to specialisation and training accorded to staff in charge of billing and finance. Users would also likely have access to high-quality scanners with email/wi-fi capability.

**Pertinent Requirements**

1. Efficiency - Corporate/Organisational users would need to process large batches of bills on a frequent and regular basis, requiring more automation tools, batch processing, and management via e-mail.
2. Security & Privacy - Due to confidentiality within an organisation, such users would require high security and compartmentalisation of bill management/access. This may require access control tools and encryption.
3. Assigning Responsibility - With a large number of users working on bills within an organisation, there should be a way to track which user made changes, requiring logging tools and possibly revision tracking tools.
4. Data Backup - Users would face severe financial impact if their data is lost due to server/hardware failure, or inconsistent network connections. Users should thus be able to backup their data in case of server/hardware failure, and some provisions for working offline made.

### 2.3.2 Professionals / High-Med Income Users

Professionals, like our primary client, are anticipated to be one major user of Bill.eGoat, as they would likely have access to higher spending and financing options such as credit cards and investments. This access would result in a multitude of bills for them to manage personally.

**Proficiency/Technical Characteristics**

Users are likely to have medium technical proficiency in IT and general web-based tools. Users would be likely to have access to smartphones and computers, while being familiar with basic application use and basic image processing.

**Pertinent Requirements**

1. Notifications - Users have busy schedules and many bills to manage, and would likely require custom notifications to alert them of pending payments or other important events.
2. Summary / Tracking - Users would benefit most from being able to track expenditure, debts, income, investments, etc. Summaries or basic tracking/analysis would assist such users to make informed choices and manage their finances better.

### 2.3.2 Households

Households are also likely to be major users of Bill.eGoat, as they have a large number of aggregated bills, which individual members may or may not be responsible for. For instance, renters may need to split and pay their utilities’ bill according to some agreed percentage.

**Proficiency/Technical Characteristics**

Users are likely to have mixed technical proficiency in IT, requiring other members to assist in managing the bills. Each household is expected to have at least one computer or smartphone, capable of interfacing with Bill.eGoat.

**Pertinent Requirements**

1. Sharing / Permissions - Household members should be able to share bills or upload bills on behalf of others. Permissions / sharing and acknowledgment tools may be required.
2. Delegating Responsibilities - Household members should be able to delegate and accept financial responsibilities to one another.

## 2.4 Operating Environment

As Bill.eGoat is a web-based application, it should be able to run on any modern browser supporting HTML 5 and Javascript, with an internet connection.

## 2.5 Design and Implementation Constraints

### 2.5.1 OCR Accuracy

Bill.eGoat would be using the Tessaract OCR library for optical character recognition, which can recognise a limited number of fonts up to a certain degree of accuracy. Although accuracy can be improved with pre-processing images and providing training texts, it is highly likely for errors in OCR to occur within each user’s experience. Hence, error-correction tools must be developed for users to overcome this technical constraint.

In addition, there is a language constraint as OCR capabilities and accuracy for non-English languages is minimal at best.

### 2.5.2 Server Space/Bandwidth/Processing Limitation

As a web application storing files on a server, Bill.eGoat would quickly run out of server space with a significantly large amount of users. Bandwidth would also be quickly taken up by users uploading large files, as well as processing cycles taken to perform processing. These constraints could be alleviated through automatic archival, shifting pre-processing workload to client devices, and/or using a paid/tiered membership system to support operating costs.

### 2.5.3 Digitization Quality

As Bill.eGoat accepts user digitized bills, the user’s image-capture device quality, skill of operation and physical/environmental conditions result in a large amount of variance in the quality of submitted digitized bills. This in turn leads to a large margin of error during processing. This constraint can be mitigated with the use of pre-processing libraries to some extent, although the onus is on the user to provide quality images. Hence, documentation and user training/tutorials/tips could be integrated into Bill.eGoat to maximise user skill.

### 2.5.4 User Platforms

Bill.eGoat aims to function on any modern browser on any web-capable device. Hence, Bill.eGoat is also limited by the same limits of these browsers. For example, deprecating support for NPAPI plugins on modern browsers means that Bill.eGoat’s design should avoid making use of such plugins to maximise compatibility. Slow corporate updates of software such as Java plugins may also pose a constraint to corporate users.

### 2.5.5 Legal Constraints

Due to the sensitive and impactful nature of financial documents, Bill.eGoat should be designed to minimise legal liabilities related to its use. Documentation and policies should be clearly defined such that users assume responsibility of Bill.eGoat’s use. However, developers should endeavour to minimise the possibility of data loss and errors in analysis.

### 2.5.6 Connectivity/Hardware Issues

Due to the possibility of inconsistent network connectivity on mobile devices, provisions should be made such that temporary loss of connectivity does not result in data loss. Loss of local or server data could also be mitigated via the implementation of redundancy through backups.

## 2.6 User Documentation

Due to Bill.eGoat’s web-based nature, user documentation/help should be delivered via tooltips, tutorials and links on Bill.eGoat itself, like an FAQ section. A simple step-through of the basic workflow should be provided online.

## 2.7 Assumptions and Dependencies

### 2.7.1 Third Party Libraries

Bill.eGoat is dependent on several third-party libraries for its OCR and image-processing capabilities. These include Tesseract OCR for optical character recognition, ImageMagick for image pre-processing, and OpenCV for logo matching.

# 3. External Interface Requirements

## 3.1 User Interfaces

### 3.1.1 Design Conventions

**Responsive UI**

As Bill.eGoat will run on any web-connected device with a modern web browser, its user interface will adopt a responsive layout in order to fit most devices. Twitter Bootstrap would be used for styling, as it responsive layout built-in to the base system.

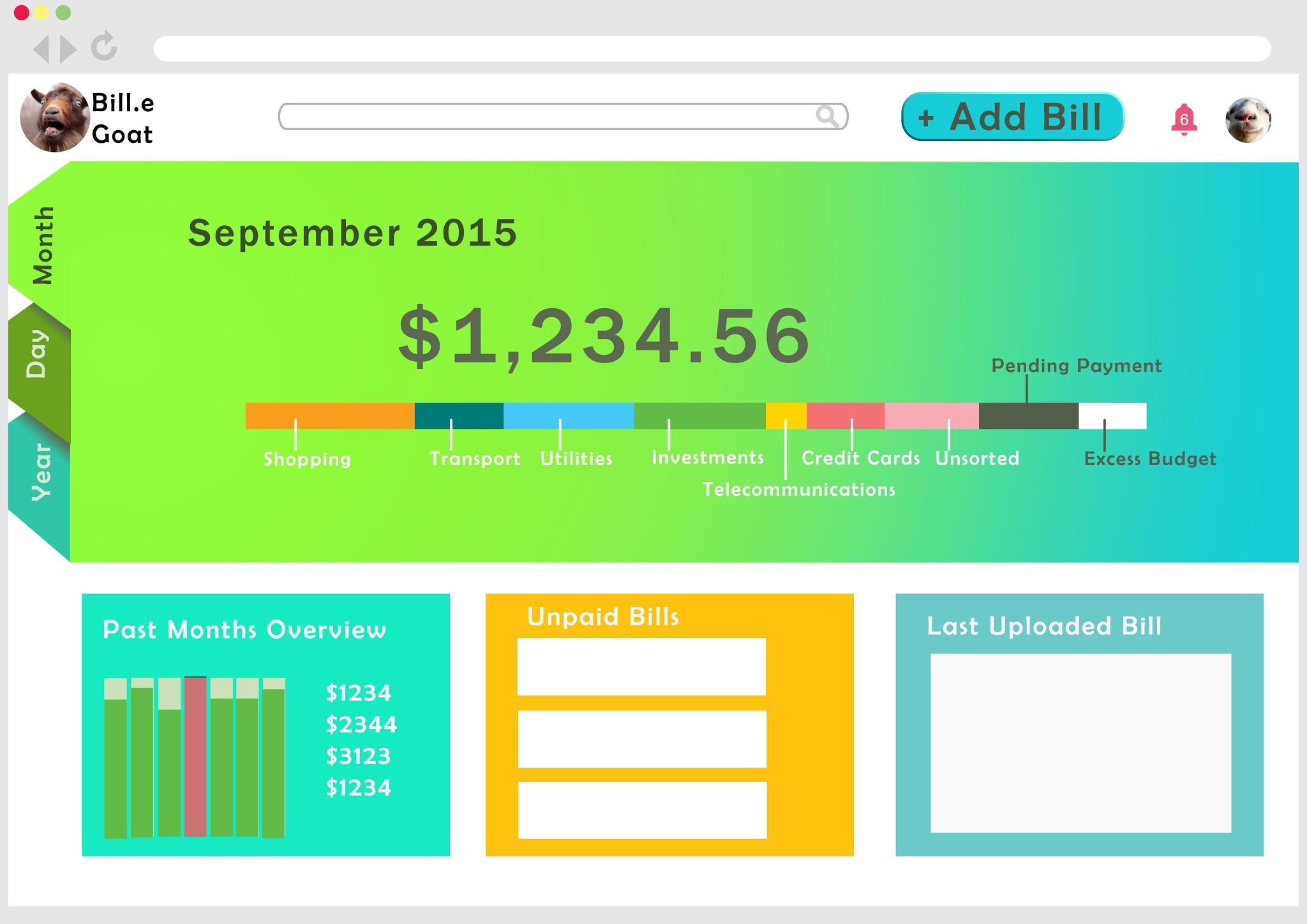
**Flat UI**

Bill.eGoat will employ a flat UI style, as minimalist graphics would not get in the way of users quickly identify the information they require at a glance. This UI style may be subverted at times in order to display an interactivity affordance (e.g, buttons), as mobile users will not be able to identify interactivity without a mouse cursor.

### 3.1.2 UI Interactions

#### 3.1.2.1 Main Application UI

The main application UI is displayed on the user’s browser, and comprises of login screens, forms, user dashboard and management pages. These would be scripted using HTML5 and Javascript / jQuery as needed. This UI links to the Bill Upload UI and the Template Manager UI directly.



#### 3.1.2.2 Bill Upload

The bill upload interface (user-facing) is simply an interface that triggers the file explorer on the client’s device to select which photos they wish to upload, or to drag and drop files onto the page. This interface is powered by HTML / Javascript. Most of the bill upload interface’s code is not user-facing, and serves to resize and upload/forward files to the server via HTTPS. This interface also links to the Image Editing UI, allowing the user to manually edit their images before uploading.

#### 3.1.2.3 Image Editing UI

The image editing UI allows users to perform basic image editing functions on their uploaded images, such as rotation/cropping, before upload. This can be achieved by various free jQuery plugins. The image editing UI links to both the bill upload interface (Save and send), and the Template Manager UI.

#### 3.1.2.4 Template Manager UI

The template manager UI allows all users to highlight and tag standardised parts of a bill, in order to create their own template. This UI will be powered by HTML5, Javascript and jQuery.

## 3.2 Software Interfaces

Bill.eGoat will interface with modern browsers on any device. Bill.eGoat should be able to operate on most web servers supporting SQL (MySQL or MSSQL) databases, PHP and libraries required by Bill.eGoat. Databases / servers will be required for account management and authentication, while a separate database / server will be required for upload and storage of encrypted bills and data, including backups. An email server will also receive incoming bills and forward them to the storage server, as well as being used for email correspondence and notifications. Identification data shared across servers and databases would be a unique account ID for each account on Bill.eGoat.

## 3.3 Communications Interfaces

Users will interface with Bill.eGoat through a web browser (HTTPS) primarily. Users may also interface with Bill.eGoat via e-mail, through email correspondence or uploading bills via email (SMTP). Uploads will typically take place via HTTPS through the web browser.

# 4. System Features

This section describes Bill.eGoat’s features, grouped by related features in the system workflow. Select user stories will also be included in these features for better rationalisation of the priority and inclusion of each feature. Features within a group are sorted by priority in descending order. Generally, functions that are high or medium priority can be expected in the production release of Bill.eGoat.

## 4.1 Data Capture/Upload

This group of functions allow users to input/upload billing data into Bill.eGoat. Bill.eGoat allows users to upload/input bills (multiple pages) from a variety of sources, through a web interface. All image formats supported by the Image Magick library will also be supported by Bill.eGoat:

**User stories for illustration:**

1. As a high flying professional with many credit card lines to handle, I want to store all my credit card bills in Bill.eGoat by scanning them on my scanner as soon as I receive them so I need not be worried about losing the original hard copy.
2. As the main breadwinner of my family, I want to easily upload the bills from my family members into my Bill.eGoat account by simply forwarding the e-bills that they forward to me from their email accounts.

**High Priority (High fidelity sources)**

1. Users can manually enter data from bills into Bill.eGoat via an extensible form interface.
2. Users can upload soft-copy PDFs/images of their bills.
3. Users can upload scanned images/pdfs of their bills.

**Medium Priority (Medium fidelity sources / Useful functions)**

1. Users can upload pre-processed (automatically/manually corrected) images of their bills from mobile scanning applications such as Cam Scanner or Doc Scan.
2. Users manually inputting data from bills can make use of autocomplete suggestions.

**Low Priority (Others)**

1. Users can upload unprocessed images of their bills from any image-capture device.
2. Users can scan to email or forward soft-copy/digitized bills (One bill per email)
3. Users can upload multiple bills (capped at 3), at any one time.
4. Users can share a syncing folder with Bill.eGoat, which accesses their files from there.

## 4.2 User Account Management

This group of functions allow users manage their account with Bill.eGoat:

**User stories for illustration:**

1. As a standard user with a bad memory, I want to be able to recover my password by receiving a link to creating a new one through my email account.
2. As a corporate user, I want to be able to remove all of my company’s data on the Bill.eGoat servers in case the company chooses to use a new system or method for organizing bills.

**High Priority**

1. Users should be able to register accounts with their own passwords and emails.
2. Users should be able to recover accounts by email.
3. Users should be able to change their passwords.
4. Users should be able to customise their notification types/frequency.
5. Users should be able to close their accounts and delete all information on the Bill.eGoat server.
6. Users should be able to add trusted email accounts to enable forwarding of bills or account recovery.
7. Users should be able to accept responsibility or shared bills on their accounts.

**Medium Priority**

1. Users should be able to add/edit/delete/close different financial accounts (I.e, bank accounts, CPF accounts, etc.)
2. Users should be able to add/accept known associates for easier sharing/tagging in future.

## 4.3 Pre-Processing Bills

This group of functions allow users to pre-process their bill images before uploading.

**User stories for illustration:**

1. As a user of the mobile version of Bill.eGoat with only 2GB of mobile broadband, I want to only upload a resized/downsampled copy of the photo of my bill so that it does not eat too much into my monthly limits.

**High Priority**

1. Bill.eGoat should automatically rotate, resize, crop, deskew and contrast images.
2. Bill.eGoat should be able to compress and downsample images before upload in a standardised format.

**Medium Priority**

1. Users should be able to undo or manually adjust/apply automatic pre-processing parameters in batches or individually.

## 4.4 Tagging and Sharing

This group of functions allow users to tag, categorise and define privacy/sharing options.

**User stories for illustration:**

1. As a SME owner, I have to deal with both business and personal bills. I want to be able to sort them into these two categories so that I can maintain separate accounts.
2. As the breadwinner in my family, I want my children to to be able to adjust their privacy settings to allow me to view their bills so I can monitor their spending.

**High Priority**

1. Users should be able to tag each uploaded bill with different categories/tags. (Preset tags for data to be used in visualisations)
2. Bill organisations and other relevant data should be automatically tagged, unless the organisation is unsupported.
3. Users should be able to edit privacy/sharing options of each bill during the tagging phase. (View/Edit)

## 4.5 Processing (Recognition) & Templating

This group of functions allow users to define bill templates and edit/tag automatically recognised data.

**User stories for illustration:**

1. As a user who tends to upload most of my bills in one shot, I want Bill.eGoat to be able to automatically extract the important details from my scanned bills so I don’t have to manually input such data for every bill.
2. As a corporate user, I will be dealing with several new organizations on a regular basis. I want to be able to add new templates to Bill.eGoat so that Bill.eGoat can start to automatically extract information from bills from new organizations.

**High Priority**

1. Uploaded bills should be automatically recognised and tagged accordingly, with reference to a centralised database of templates.
2. Users should be able to edit bill data and tags before confirmation, or skip this step.
3. Users should be able to use a drag-to-highlight interface to design their own template for unsupported organisations, highlighting and tagging where billing information occurs on their bill. This template should be saveable and reusable in future.

**Medium Priority**

1. Users should be allowed to edit and overwrite official preset organisations with their own templates if they wish to.

## 4.6 Data Analysis / Dashboard

This group of functions allow users to analyse or summarise their billing data over a time period, as well as get a sense of their current financial position through a dashboard interface.

**User stories for illustration:**

1. As a simple user, I want to be able to quickly find and view my outstanding amounts to be paid without too many steps.
2. As an absent minded user, I want to be able to double check what I last loaded into Bill.eGoat.

**High Priority**

1. Bills should be automatically sorted, with options for sorting by billing organisation, dates (Date Submitted / Date Due / Billing date / Date Cleared), billing amount, user defined tags and filename.
2. Users should have access to a dashboard with their financial summary for the week/month/year (Toggleable).
3. Users should be able to see a summary of their income and expenses for a time period at a glance.
4. Users should be able to access their last bill uploaded easily from the dashboard.

**Medium Priority**

1. Users should have access to data visualisations and summaries of their bills over a specified time period.
2. Users should be able to access breakdown summaries / representations of their income / expenses.
3. Users should be notified of unverified pending bills, if users skip manual tags and rely solely on the automatic recognition. This would also occur if bills are sent via e-mail.

## 4.7 Bill Retrieval & Management

This group of functions allow users to edit individual bills’ data, permissions and flags, as well as search, revision control, annotation/redaction and notification tools.

**User stories for illustration:**

1. As a standard user, I want to be able to download and print out a copy of a bill of which the original has been misplaced in case I need to submit it to someone.
2. As someone who lives hand to mouth and pays off bills individually whenever I can, I want to be able to mark specific individual bills as having been paid so that I no longer need to track them in my due bills.

**High Priority**

1. Users should be able to edit, delete, void or archive (manual or automatic) each bill.
2. Users should be able to see revision histories and access logs of each individual bill, reverting at least up to 3 revisions.
3. Users should be able to grant / revoke permissions on each bill.
4. Users should be able to edit any bill’s fields or tags.
5. Users should be able to search for or view a sorted list of bills using keywords, tags, dates or other recorded data.
6. Users should be able to flag a bill as cleared, uncleared, or outstanding balance.

**Medium Priority**

1. Users should be able to set custom notifications/flags for a category / individual bills, such as an alert if a bill is due to be paid, or if there are any excess charges.
2. Users should be able to download the server-side copies of their bills in a sorted zip directory. (Option of how bills would be sorted, by date or billing organisation, or both).

**Low Priority**

1. Users should be able to annotate / redact bills using a similar interface to the templating interface, be tagged as a variant of the original bill, and shared separately.
2. Users can opt for e-mail notifications in addition to application notifications.
3. Users should be able to sync the server’s bill directories with a local copy on their device, such as via Dropbox or other syncing solutions.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

### 5.1.1 Client-side Performance

**Download and Processing**

Bill.eGoat should take no longer than 30 seconds to download and execute the necessary code for processing client-side, to avoid excessive data use (for mobile users) and user frustration. Optimally, download and execution should take less than 10 seconds on a standard LTE connection. As users may be on less than optimal mobile networks or other connections, this constraint may be mitigated by the use of caching or shifting more processing server-side.

**Upload Performance**

Similarly, uploading one page of a bill (one image) should take no longer than 10 seconds. Upload timing would be almost impossible to approximate if users are allowed to upload files of varying formats and sizes. Hence, Bill.eGoat requires a library client-side to pre-process and downsample images before upload, which should take no longer than 30 seconds.

### 5.1.2 Server-side Performance

Server-side processing should not take longer than 30 seconds per page of each bill on average, depending on server load. To maximise user productivity, users should be able to select privacy settings and other annotations for each bill as the bills load, with a progress bar. Page content could be dynamically loaded as it finishes.

## 5.2 Security Requirements

Bill.eGoat has to have high security and protection of personal data due to the nature of bills, requiring the use of encryption and user authentication. Account recovery or dispute cases would necessitate submission of proof of identity, as bills previously uploaded would include the user’s identity as well. Developers may consider the use of 2nd factor authentication (2FA) as well, if commercially viable.

## 5.3 Software Quality Attributes

### 5.3.1 For Developers

Bill.eGoat has to satisfy common code quality metrics for developers, with clean, compartmentalised code written. Code should minimise coupling, allowing developers to use each other’s modules easily. Code should be written for robustness and correctness of high priority features first, then extendible for medium and low priority features.

### 5.3.2 For Users

Bill.eGoat aims to maximise efficiency for users by minimising steps needed to process a single bill. However, error correction capabilities are second priority, allowing users to heavily edit a bill if need be. In general usability terms, Bill.eGoat should provide intuitive interfaces in alignment with current conventions that most users should be familiar with. Bill.eGoat should be robustly tested on one set of documents/use cases before moving on to other use cases - essentially, depth over breadth of supported features.

## 5.4 Operating Principles

Corporate partners may access Bill.eGoat’s template preset system to set templates for their bills, as verified by our administrators. They may also require tools to push bills to users through Bill.eGoat.

Administrators, which are the developers, need to be able to determine account hierarchy levels of all other users, as well as preset templates as and when there is a demand for them.

Normal users may require tiered services, to cater to different needs.

# 

# 

# Appendix A: Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Bill | An invoice detailing payment terms from a seller to a buyer, for products or services. This includes phone/internet/utilities services, credit card/bank/investment statements. |
| Bill.eGoat | A web-based system that semi-automates sorting & analysis of digitised bills |
| Developer | Programmers or designers of Bill.eGoat |
| OCR | Optical Character Recognition |
| NPAPI | Netscape Plugin Application Programming Interface, a cross-platform plugin. |
| Digitize | Conversion of a physical bill into digital image form via photographic or scanning technology. |
| Soft-copy | A copy of a bill in a digital text format |
| PDF | Portable Document Format, a format which can include multiple pages of soft-copy bills or digitized bills. |
| Dashboard | Home area of Bill.eGoat with financial summaries for the user |
| CPF | Central Provident Fund, a mandatory savings fund for citizens and permanent residents of the Republic of Singapore |
| Dropbox | A web-based file hosting service, with a free tier for low-usage customers. |
| Image Magick / OpenCV | Open-source libraries for image editing, available in several languages. |