SRS - eBanking Web App

Group Four
Anthony DeDominic <dedominica@my.easternct.edu>
Courtney Combs <combsco@my.easternct.edu>
Jeremy Drexler <drexlerj@my.easternct.edu>
Kevin Bailey <baileyk@my.easternct.edu>

May 6, 2016

Contents

i. Revisions

Since the documents for this group are version controlled and written in a plaintext, human readable, format, changes to the SRS can be found at https://github.com/adedomin/CSC385-Documents/commits/master

Commits clearly show the sections that were added to the SRS, changed, or even removed.

1. Introduction

The e-bank System is a database driven website designed to support online banking operations using the basic functionalities, such as login/logout, view account, transfer funds, pay bills, utility, and security. The system design is developed through a series of processes to ensure all customers' needs are meet when banking online. More specifically, customers who have the online banking account should be able to login the system by entering a valid username and password. If the username and password is valid the customer can view their account. View account allows a customer to see up to date balance information on deposit (saving/checking), credit card, debit cards and any other bank information. The account also allows customers to view their transaction history up to a maximum of 180 days. Transfer funds allows customer to transfer funds between authorized personal accounts. Requested transfer takes place immediately or at a selected future date specified by customer. Customers are allowed to view and/or terminate pending transfers. The customer can use online pay bill service to pay bills by debiting their account. This payment is made to payee corporations that the customer has registered with online banking by using the registered bills. With new payee corporations that the customer has not registered, this payment can be made immediately or at a later date. Utility allows customer to change password, secure delivery contact information, and choose the types of security questions. This allows the customers who forget their password be able to reset it through their email but to also ensure it through a secure system. Customers can also change the online profile and personal information. When the customer is done using e-bank they can simply logoff. If the customer stays on for 20 minutes

while the site is unused then the system will automatically log the customer off due to security reasons.

1.1. Purpose of the System

The Purpose of e-bank is to help customers make banking easily available when a customer needs to access their funds. Some of the problems e-Bank faces:

- Customer(s)
  - Validation of the customer's username and password and ensuring security to discourage those who seek other customer's information.
  - Transfer funds from the customer's account to another person's account. Valid banking information must be provided or the funds will not transfer.
- End-User(s)
  - Valid information for transferring funds to an outside source will allow the end user to accept the funds coming in.
  - Transaction occurs the bank takes out too little or too much by mistake will cause customer accounts to be off.
- Developer(s)
  - Too small of a database.
  - Creating a database that is successfully at authorizing usernames and passwords to each customer.
  - Database overload causes the system to go offline.
- Concept statement- To overcome these problems it is the developers job to ensure that the proper sized database, the validations to username/passwords, the transactions are made to the design standard as well as the customer's standards. This will be done so by with each section of the design system being made proper checks are done then showed to the customer.

1.2. Scope of the System

This project will consist of creating a marketable online banking system based upon customers' needs in the modern era of increasing technology demands in the business world. This system has a front-end for user interaction and a back-end for employees and web developers updating and conducting maintain on the site. The project will be completed by April 29, 2016. Modules of e-Bank will include username/password access, view account, transfer funds, pay bills, utility, and security measures. This is a great way for Banks to earn money, and a way to encourage customers to use online banking by accessing their funds quicker, pay bill on time, and the system is easier than ever before.

1.3. Objective and Success Criteria of the Project

e-Bank makes it easy for users to bank online without the hassle of physically going to a bank. Although e-Bank is easier to use, customers must know that if someone gets a hold of their username, passwords, or both could compromise their banking information. Therefore, extra firewalls, security questions, and personal information will be ensured for this system to not allow any unauthorized users to compromise any customer information.

e-Bank will be web-based, so users will be able to collaborate without being in the same physical location. It will provide a simple interface that will function across platforms and will be understandable by all e-Bank users. Whenever possible, its features will remain flexible, so management of a project may be configured by users to meet the team's specific needs. A large section of e-Bank's flexibility will be the availability for users to make the necessary banking transaction needed and the system's transactions can be used into subgroups in order to help pay bills or even keep track of spending money. This will allow users to keep organized in their daily financial tasks.

2. System Context

2.1. System Concept

The automated banking system shall utilize a three-tier architecture, consisting of:

- A front-end web interface to which multiple users can connect to through an HTTPS communication over the Internet.
- A middleware web server that handles authentication and data management.
- A back-end database which stores data concerning users and transactions.

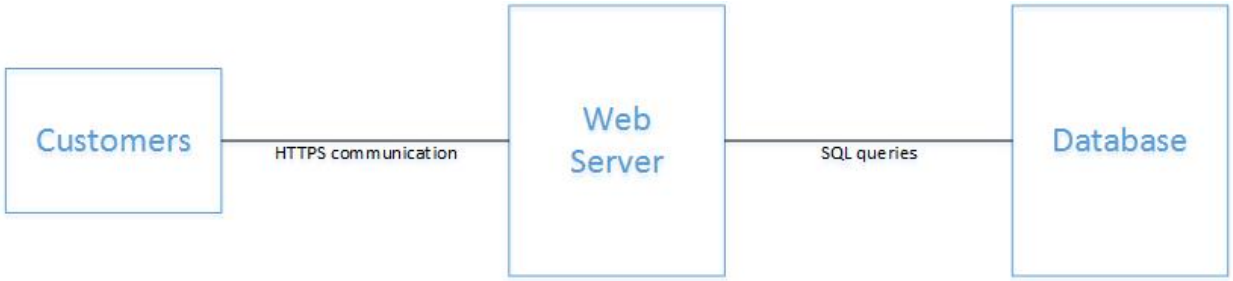

Figure 1: 2.1.1.[1]

[1] A high-level system concept model.

2.2. Current System

N/A

2.3. System Overview

The automated banking system is itself in the domain of a larger banking system. Excluding functions of the banking firm irrelevant to the automated banking system.

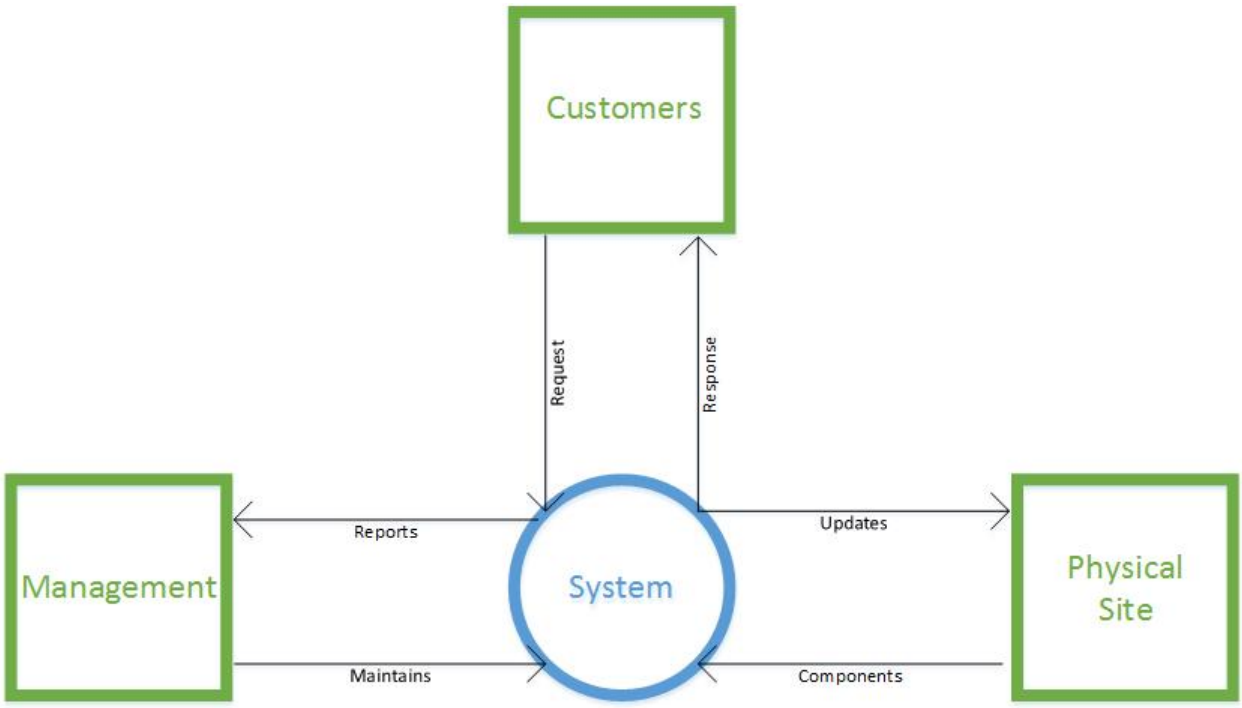

Figure 2: 2.3.1.[2]

- The management to whom the system reports. This party is responsible for system maintenance and evolution.
- The customers to whom the system services. At a fundamental level, the users communicate with the system through a typical HTTPS connection. The user initiates the communication by sending a request to the system, to which the system responds accordingly.

- The physical site on which the database and/or web server is located. All system components pertaining to the:

[2]System context diagram of the automated banking system. Highlights the outside forces affecting the system.

- Hardware and software
- Utilities
- Space

2.3.2. Web Interface Overview

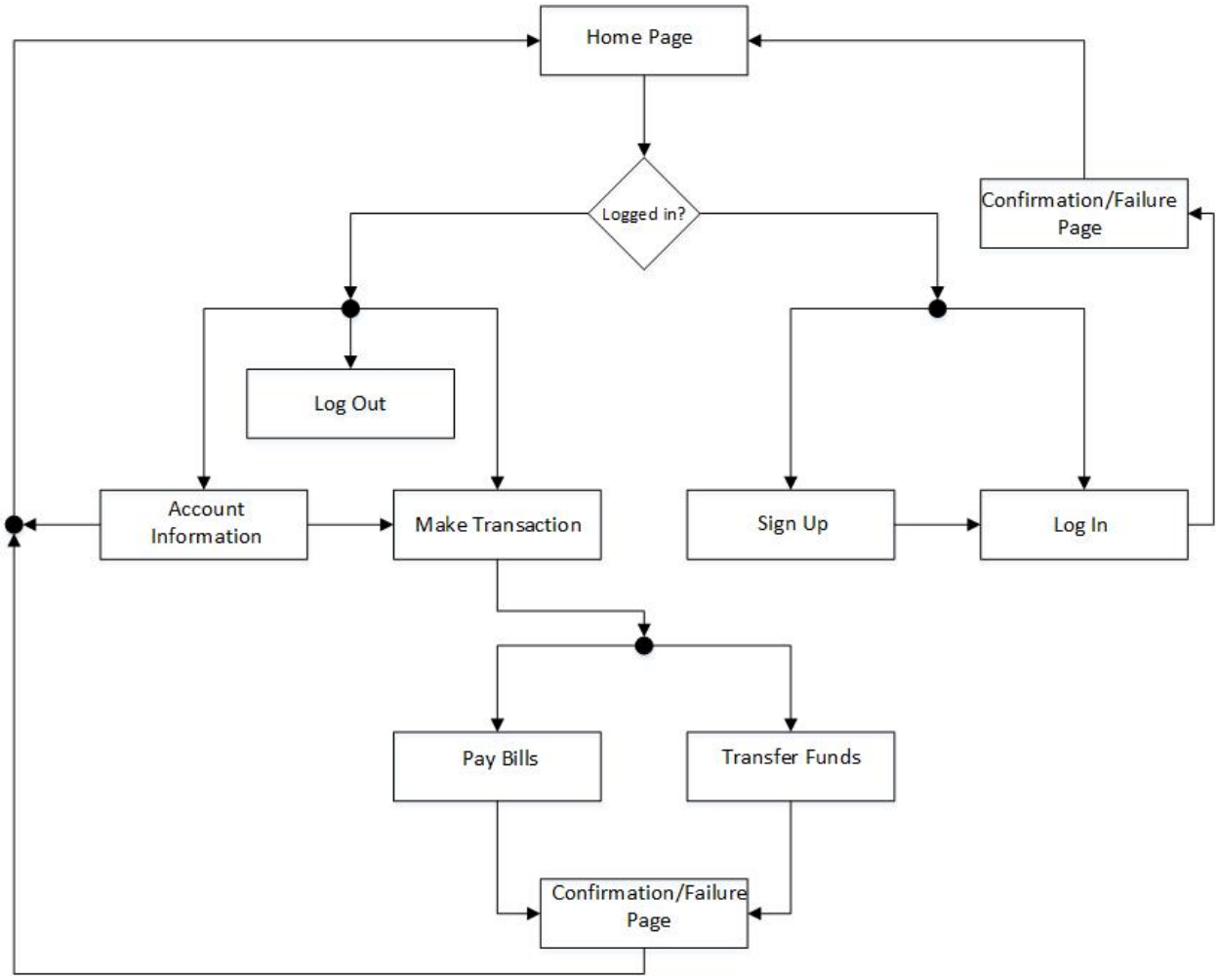

Figure 3: 2.3.2.1.[3]

This is a representation of the tasks a user can complete in a given session with the banking system. The diagram assumes the user begins on the home page, in which they may or may not be logged in. The home page shall display different content depending on the login status of the user.

[3]Clickstream diagram for the system's front-end interface.

If logged in, the user shall have an option to make a transaction, view their account information in more detail, and log out of their current session. Should the user attempt to make a transaction, they shall have the option to pay current bills or transfer funds. A page appearing after the transaction will alert the user to whether or not the transaction was successful before redirecting them back to the home page.

3. Requirements

3.1. Functional Requirements

As this web application deals with highly sensitive customer data, it shall log all actions so as to assist in recovering from wrongful events.

3.1.1. General logging

1. The web application shall log all requests made to the application.
    1. The web application will take a particular focus to log transactions:
        1. Withdrawls, the removal of money from an account.
        2. Deposits, the addition of cash into an account.
        3. Transfers, Inter-account movement of cash.
        4. Bill pay, payment of bills to sponsered partners.
    2. The web application shall also log all HTTP requests.
        1. The web application shall log, the full path and the HTTP verb (e.g. GET /some/path).
        2. The web application shall log, the IP of the requestee.
        3. The web application shall log, if the user is authenticated; if so record this user's name.
        4. The web application shall log, the time of the request, likely in ISO 8601 format, TBH.
        5. The web application shall log, all java errors, exceptions or unexpected errors in the business logic.
        6. The web application shall log all unauthorized requests.
    3. The web application shall log all changes to the user's account.
        1. Such changes as, if a user changes their login name.
        2. The web application shall log when a user changes their email.
        3. The web application shall log when a user a user does password recovery.
        4. The web application shall log when a user changes their billing a address.

5. The web application shall log when a user requests or changes their notification settints.
6. The web application shall log when a user adds an account.
7. The web application shall log when a user removes an account.
8. The web application shall log when a user adds a recurring job (cron, etc).
9. The web application shall log when a user removes a recurring job (cron, etc).

3.1.2. Storage of Logs

2. The web application must store logs both redundantly for security reason, but ensure the logs are easily readable to both humans and batch processing software as well.
    1. The web application shall use the storage offered by the hardware it resides on.
        1. The web application shall write to the disk provided by the hardware platform it executes
        2. **UNLESS** the web application is on a PaaS architecture that does not provide any disk access.
    2. The web application shall store logs on remote systems such as a database and a log server
        1. The web application shall store logs off-site, where it will be kept redundantly, will be compress and where the logs will be rotated daily.
        2. The web application shall store logs in MongoDB to assist in the performance of batch analysis of the logs and to allow for multiple concurrent readers of the log.
        3. The web application shall conform to a schema for both plaintext form and database form

3.1.3. Notifications

To ensure the users security, the application should make available to them a way to be notified when certain actions occur. Notifications can be sent over services like email, texts, phone notifications or for advanced users, custom URIs. Notifications should be useful to the user and should only expose a part of what is logged.

3. The web application shall offer a part of what it logs to be sent to a user in a variety of forms.
    1. The web application shall provide logging information pertaining to the following.

1. The web application shall allow users to be notified to all changes to their accounts. (as shown in functional spec 3.1.1.3.).
2. The web application shall allow users to be notified to all transactions logged. (as shown in functional spec 3.1.1.1.)
3. Users shall be able to remove notifications from the web app.

2. The web application shall store pending notifications in form of a FIFO queue.
    1. The queue will include all notifications that the user has not recieved.
    2. If the user has no notifications, the queue shall always be empty.
    3. As th euser reads the notifications they shall be removed.
    4. notifications shall only have a life of 24 hours.
3. The web application shall provide the following mechanisms for notifying the user.
    1. The web app will allow users to specify their registered emails as a place to send their notifications.
    2. The web app will allow users to retrieve RSS feeds of their recent alerts.
    3. The web application will allow users to use our banking notification app to retrieve notifications.
    4. The web application will also allow users to specify a website url to call when notifications become available which will allow users to use their own notification services or a third party notification service.
        1. all notification data will be sent via a POST or PUT and only over a secure connection.

3.1.4. APIs and Interactions

Before beginning the spec, it is wise to first indicate the routes that users, admins and the automated bank system will use to interact with the web application.

Verb	Path	Role
GET	/	NONE[4]
GET	/login	NONE
POST	/login	NONE
GET	/logout	NONE
GET	/signup	NONE
POST	/signup	NONE
GET	/api/v1/user	USER[5]
POST	/api/v1/user	USER
POST	/api/v1/user/new	USER

[4]NONE role means the user does not need to be logged in.
[5]USER role means the user is a regular, authenticated user.

Verb	Path	Role
GET	/api/v1/user/[name]	ADMN[6]
POST	/api/v1/user/[name]	ADMN
GET	/api/v1/user/[name]/reset-pass	NONE
GET	/api/v1/user/[name]/reset-pass/[key]	NONE
GET	/api/v1/accounts	USER
POST	/api/v1/accounts	USER
POST	/api/v1/accounts/new	USER
GET	/api/v1/accounts/[id]	USER
DELETE	/api/v1/accounts/[id]	ADMN
GET	/api/v1/accounts/[id]/history	USER
GET	/api/v1/accounts/[id]/card	USER
POST	/api/v1/accounts/[id]/card	USER
POST	/api/v1/accounts/[id]/card/new	USER
POST	/api/v1/accounts/withdraw/[id]	BANK[7]
POST	/api/v1/accounts/deposit/[id]	BANK
GET	/api/v1/external	USER
POST	/api/v1/external/new	USER
POST	/api/v1/external/[id]	USER
DELETE	/api/v1/external/[id]	USER
GET	/api/v1/transfers	USER
DELETE	/api/v1/transfers/[id]	USER
POST	/api/v1/accounts/[id1]/to/[id2]	USER
POST	/api/v1/accounts/[id]/to/external/[id]	USER
POST	/api/v1/external/[id]/to/accounts/[id]	USER
GET	/api/v1/bill-pay	USER
POST	/api/v1/bill-pay/new	USER
POST	/api/v1/bill-pay/[id]	USER
DELETE	/api/v1/bill-pay/[id]	USER
GET	/api/v1/bill-pay/[id]/history	USER
GET	/api/v1/notifications	USER
POST	/api/v1/notifications/new	USER
POST	/api/v1/notifications/[id]	USER
DELETE	/api/v1/notifications/[id]	USER

[6] ADMN role means an administrator role.

[7] BANK role is a special role to syncronize the bank's internal state with the web app.

3.1.4.1 Logins

The login paths, /login, /logout, etc, are paths where users can initiate a session with the web application.

1. The Application will serve forms created by server side templates to any of these paths.
    1. The /login form will be a simple two field form which will ask the user for a username and a password.
    2. The /login form will also offer a link to sign up for an account, a reset password link or a link to contact customer support.
    3. The /logout link will return a message indicating the user successfully logged out.
    4. The /signup form will return a form object which will ask a user for the following information shown in the table below:

Field	Type	Description
first	String	The user's real, first name.
last	String	The user's real, last name.
email	String	The user's email, also their username.
phone	String	the user's phone number.
password	Password	The user's password.
verifyPass	Password	The user's password repeated to ensure correctness.
streetAddress	String	The user's street address.
city	String	The city the street address resides in.
state	Combobox	the state the user resides in.
Other	unknown	other information may be requested.

3.1.4.2 User

1. The user routes, starting with /api/v1/user shall allow the user to do the following.
    1. When the user asks for /api/v1/user, he shall recieve his account information; the user's password is **NOT** returned.
    2. When the user sends part of, or a completely new model, as a POST request to the above route, the web application shall apply the fields, that are allowed to be changed, to their user profile and the changes will be saved into the database.
    3. the /api/v1/user/new route will allow users to create new users in an automated fashion. It takes the same data as the /signup form.
2. If the user is an admin, the admin shall be allowed to do the following.

    1. See **ALL** data about any given user, minus that user's password.
    2. Change any field in the user's profile except for their password.
3. If the user needs to reset their password, they shall make a query to do so at /api/v1/user/[the username]/reset-pass.
    1. The user's email will be notified of the request and will contain a link where a user can go to and change their password.
    2. The user, once navigated to the link will be given a form to change their password; the form will contain password, and a duplicate password field to verify they are both the same.

3.1.4.3 Accounts

This is the most complex portion of the web application. This api allows for many actions, from retrieving the the status of accouts, but also transaction history (up to 180 days), inter-account transfers, withdrawls and deposits.

1. The system shall return all the user's Accounts when the user accessess /api/v1/accounts
    1. The web application shall allow for a user to create a new account through /api/v1/accounts/new
    2. User's shall be allowed to fetch the information of just one account via /api/v1/accounts/[id].
    3. Only a admin can remove accounts as the value of the account needs to be migrated in some way. The removal will delete the Database entry for that account including the credit cards and so on.
    4. The user shall be able to transfer a sum amount from an account to another account.
        1. The system shall not transfer so much that there is not a negative balance.
        2. The system shall allow a user to define a date and time to delay the transfer. If the balance is insufficient the transaction will fail.
        3. The system shall allow a user to see all transactions and allow said user to cancel or delete them at any time.
    5. The user shall be allowed to issue a bank card (debit) for any and all accounts the user has open.
    6. The user shall be able to delete cards assuming there is no outstanding balances or overdrafts.

3.1.4.4 External Accounts

This section describes the interaction between internal accounts and external accounts at other organizations.

1. Users shall be able to add external banking accounts
    1. the user shall be able to create external bank accounts to be tracked by providing the account and routing number.
    2. The user shall be able to delete these accounts.
    3. The user can also transfer money to these accounts, or from these accounts if so desired.

3.1.4.5 Bill Payer API

1. The web app shall provide corporate partners to allow for automated bill paying.
    1. Users shall be able to select from providers and schedule for automatic bill paying.
    2. Users shall be able to retrieve the list of providers the user is automatically paying.
    3. Users shall be able to delete these automatic bill paying at any time.

3.1.4.6 Notifications

1. The web application shall, for convenience, offer notifications for the user.
    1. The user shall be able to set up a new notification service.
        1. The user shall be able to pick one of the following methods for notification; all notification types will have an associated queue of unread notifications. Once the queue is sent to the end-points, the notification is removed from the system.
        2. The user shall be able to choose to be notified via email.
        3. The user shall be able to choose to be notified via text messages. (MAY NEVER BE IMPLEMENTED DUE TO BUDGET AND OTHER CONSTRAINTS)
        4. The user shall be able to choose to be notified via a notification app built for the web app.
        5. Users wishing to use third party notification services shall be able to specify a callback url that the webapp will send the notification through.

3.2. Nonfunctional Requirements

The Application has many security and operational concerns. Users can potentially lose money through insecurities so it's critical the application is built securely and to allow for operations to assist users who may be defrauded.

3.2.1. General Security

Security is paramount for the application. The application must make use of cryptographically secure transportation like TLS. To ensure our user's security, it is required that architecture, such as the database, must be password secured or behind a firewall preventing it from listening to the outside world.

The web application must make use of framework technologies. Frameworks are built and designed around securely and easily serving content over the web. Security concerns pertaining to the framework are also handled externally saving time.

3.2.2. Scalibility

The application should be designed to be capable of scaling. The Application should be capable of being load balanced to allow for easy scalibility. To ensure the application can be load balanced, the application should rely on external servers for storage and databases. So The application, or applications, shall connect to remote storage pools and databases for their persistent storage needs. This will allow all the running instances of the app to share their states with one another.

Databases, like MongoDB, generally support mechanisms like sharding to allow for scaling up database requirements. Because of this, MongoDB was chosen for this application. MongoDB also has the ability to serve as a general purpose filesystem using GridFS. This further supports load balancing the application among several machines.

This requirement also plays into high availability, since a failure in one sub system will not cause total failure. This is because the data models and the server side logic is distributed.

3.2.4. Continuous Integration and Deployment

The system's source code must be managed in a way that allows for automated building tools like Jenkins to generate deployable and versioned binaries. Binaries should be managed in a way that allows for automatic deployment to production systems.

This implies all code and configurations need to be hosted in a remote source code management system such as a git repository hosted on github. Binaries must be sendable to a server with accompanying versioning information. Binaries can be sourced from this binary repository by deployment tools like ansible or puppet. From there, ansible or puppet can install and run the necessary configurations and binaries.

3.2.3. Operations

There should be a portal or external views and applications that allow operations teams to view and modify portions of the database. The system should also monitor anything that can and/or will go wrong so operations should be able to quickly recover from outages.