# **Software Object-Oriented Analysis**

*Help Me! Laurier*

Vaughan Hilts (120892740)   
Brandon Smith (120201510)   
Colin Gidzinski (120663460)

Table of Contents

[Introduction 5](#_Toc403050145)

[Identifying Concrete Objects 6](#_Toc403050146)

[A brief notice on actors 6](#_Toc403050147)

[Entities 6](#_Toc403050148)

[User 6](#_Toc403050149)

[Submission 7](#_Toc403050150)

[Question 7](#_Toc403050151)

[Answer 8](#_Toc403050152)

[Notification 8](#_Toc403050153)

[ProfileQuestionEntry 9](#_Toc403050154)

[AbuseReport 9](#_Toc403050155)

[AuthenticationToken 10](#_Toc403050156)

[Boundaries 10](#_Toc403050157)

[LoginPage 10](#_Toc403050158)

[GoogleAuthenticationButton 10](#_Toc403050159)

[WelcomePage 11](#_Toc403050160)

[DismissButton 11](#_Toc403050161)

[ParentFrame 11](#_Toc403050162)

[ViewNotificationsButton 11](#_Toc403050163)

[HomeButton 11](#_Toc403050164)

[EditProfileButton 11](#_Toc403050165)

[HomePage 12](#_Toc403050166)

[AskButton 12](#_Toc403050167)

[ViewQuestionsList 12](#_Toc403050168)

[NotificationPage 12](#_Toc403050169)

[QuestionsForYouList 12](#_Toc403050170)

[AnswersForYouList 12](#_Toc403050171)

[SubmissionPage 13](#_Toc403050172)

[TitleTextBox 13](#_Toc403050173)

[DescriptionTextBox 13](#_Toc403050174)

[FormattingWidget 13](#_Toc403050175)

[SubmitButton 13](#_Toc403050176)

[ViewTopicsPage 13](#_Toc403050177)

[ViewQuestionPage 14](#_Toc403050178)

[SubmissionCard 14](#_Toc403050179)

[ReportQuestionPage 14](#_Toc403050180)

[ProfilePage 14](#_Toc403050181)

[Control Objects 15](#_Toc403050182)

[Server Control 15](#_Toc403050183)

[Authentication Control / OAuth Control 15](#_Toc403050184)

[NotificationControl 15](#_Toc403050185)

[QuestionControl 15](#_Toc403050186)

[AnswerControl 16](#_Toc403050187)

[Activity Diagram 17](#_Toc403050188)

[Use Case Sequencing Diagrams 18](#_Toc403050189)

[Use Case Sequence: Asking a question 18](#_Toc403050190)

[Use Case Sequence: Answer a question 18](#_Toc403050191)

[Use Case Sequence: Check a question 19](#_Toc403050192)

[Use Case Sequence: Login and setup 19](#_Toc403050193)

[Use Case Sequence: Check notifications 20](#_Toc403050194)

[Use Case Sequence: Distribute a question 20](#_Toc403050195)

[Use Case Sequence: Push an answer 21](#_Toc403050196)

[Use Case Sequence: Update profile 21](#_Toc403050197)

[Use Case Sequence: Reporting a question 22](#_Toc403050198)

[Use Case Sequence: Connection Down 22](#_Toc403050199)

[State Diagrams 23](#_Toc403050200)

[User 23](#_Toc403050201)

[Submission 24](#_Toc403050202)

[Notification 25](#_Toc403050203)

[AbuseReport 26](#_Toc403050204)

[AuthenticationToken 27](#_Toc403050205)

*This page is intentionally left blank.*

# Introduction

In the proceeding sections we will flesh out the concrete entities, boundaries, controllers and application specifics in a manner that gives greater insight on the more granular operations of the application.

To do this, we must define all the objects we will be using at a high level and the entire spectrum of objects will be using throughout the application. Below, you will see a breakdown of the different pieces of the application, how they relate, and how they will interact with each other.

While some attributes on various entities and objects will be provided, this can be thought to be only a high level overview of everything included in the application. For more detailed outlines, one must look at the *Design* documentation. The focus will be on the broad identified objects in the proceeding documentation.

*The Table of Contents can be used to see a good hierarchy of the total application. It is recommended to give it a brief look before continuing to get a good scope of the project.*

# Identifying Concrete Objects

## A brief notice on actors

**The various user actor variations explained:** The requirements and some of the documentation will directly refer to the actors **Curious User and Experienced User.** These are both represented as a persistent **User** entity within the system application, but to make it clear what they represent, they have been separated in the use cases. A user can be both with one **User** entity but only one at a time.

## Entities

Below, you will find a brief overview of every entity in the application without too much detail to save time in getting a general overview. **Note:** Each entity which is similar to another (read: inherits) is placed a subheading to group them together. For a very loose hierarchy of this, see the table of contents.

**Note:** Where applicable, a quick brief overview of potential attributes will be listed.

### User

The persistent representation of our end user in the application is shown as an actor throughout the application and use cases. The user will typically submit **Questions**, **Answers**, update profile data and generate the content for the application. Users may also generate various types of reports through specific actions, as described below. Users are composed of names, descriptions, and important profile data. They also contain lists of questions and answers that belong to them.

|  |
| --- |
| **User** |
| + UserId  + QuestionsAsked  + AnswersProvided  + State  + Score |

### Submission

A submission is a piece of content on the application. It is typically modelled as a **Question** or **Answer**, but it could be other things down the road such as comments. For those familiar with internet forums, it can be seen analogous with a forum post in many cases. They contain information, usually textual that describe useful information to be conveyed to a **User.**

|  |
| --- |
| **Submission** |
| + Id  + AuthourId  + Body  + Status  + Score  + Date  + CategoryId |

#### Question

A **Question** is a specific piece of content a **Curious User** actor will ask, which is tied to a **User** entity. For each question posted by the application by a single user, there will be a single entity of the type **Question** mapped to it. Questions can store bodies, scores and states.

|  |
| --- |
| **Question : Submission** |
| **+** Title |

#### Answer

An **Answer** is a specific piece of content (**Submission)** an **Experienced User** actor will usually submit. These are also tied to a **User** entity, similar to the previous question entity. Each answer contains a score, body and a state. In fact, the only distinguishing feature between a **Question** and **Answer** is the various states and business logic that can be applied to them.

|  |
| --- |
| **Answer : Submission** |
| + ParentQuestionId |

See *State Diagrams* to get a better idea of the various states the different entities can be split into.

### Notification

A **Notification** is an entity that represents the state for a given possible even in the application. Typically, when a new **Answer** or **Question** has been pushed to the **User**, they will receive an updated entity that represents this particular entity. The **Notification** entity encompasses all the basic information a **User** needs to know about the event at a glance, without actually interacting with the said event.

|  |
| --- |
| **Notification** |
| + Id  + Type  + Description  + Title  + Date  + UserId  + Status |

### ProfileQuestionEntry

A **ProfileQuestionEntry** represents and stores information that a **User** has voluntarily provided on their profile page in the application. This is persistent information the application will use to assess items such as submission suitability and what kind of **Notification** entities are generated for the user.

|  |
| --- |
| **ProfileQuestionEntry** |
| + Id  + Title + Description + Value |

### AbuseReport

Represents an abuse case raised against a **Question** or **Answer**, often generated by the use cases “**Reporting a Question”** and “**Reporting an Answer”,** that is persistent and required to be handled by the application. Tied to a specific **User** and stored externally from the application to be handled. Actions performed due to an **AbuseReport** will affect both this entity and the **Question** or **Answer** it has been raised against.

|  |
| --- |
| **AbuseReport** |
| + Id  + Reason  + Description  + ReporterId  + SubmissionId  + Status |

### AuthenticationToken

The **AuthorizationToken** is a very specific type of entity that is only used for storing authorization requests and tokens for OAuth providers. We require keeping this around to be able to authenticate a **User** with their WLU account via OAuth providers like Google. For more information regarding tokens, check the official Google API:

<https://developers.google.com/accounts/docs/OAuth2>

|  |
| --- |
| **AuthentcationToken** |
| + Id  + OwnerId  + Hash  + ExpiryDate  + IssueDate |

## Boundaries

In our analysis, boundaries are considered elements that our actors will interact with. For the most part, this is simply the **Curious User** and **Experienced User.** Boundaries typically are things like frames, pages, buttons, and other elements a user will touch, slide, view and transition into.

**Note:** For brevity sake, the suffix “Boundary” has been omitted on most boundaries in the proceeding sections.

### LoginPage

A generic page used to display information about logging in to the **User**, contains many sub boundary objects which may be important.

#### GoogleAuthenticationButton

A button that is used to initiate the authentication step of the **Login and Setup** use case. When pressed, users will be presented with authentication options.

### WelcomePage

A page that displays introductory information regarding the application to a first time **User** to help them get accustomed. If this page is shown, it is because the use case “**Login and setup”** has just been invoked.

#### DismissButton

A button that is used to dismiss the dialog, WelcomePage, and let the “**Login and setup”** use case continue on in flow, leading to the **LoginPage.**

### ParentFrame

A frame that encompasses various boundary controls, mostly pages. This boundary control is almost always visible during the application, except when then the **WelcomePage** and **LoginPage** are within view. It houses buttons such as…

***Header***

#### ViewNotificationsButton

When this button is touched, the use case “**Check notifications**” will be invoked and the **User** will be able to check their notifications.

#### HomeButton

When this button is touched, the active page in the application will switch to the **HomePage.** No particular use case will be invoked.

#### EditProfileButton

When this button is touched, the active page in the application will switch to the **ProfilePage.** Changes to specific **User** profile information can be changed here.

### HomePage

A page that has some basic tools and utilities for viewing questions, asking them and sorting question content within the application is shown here. More interestingly are the controls housed on this page.

#### AskButton

When touched, the use case “**Ask a question”** is triggered and the user will be put through the flow of events.

#### ViewQuestionsList

When interacting with this control, the application will transition and allow viewing of previous questions that have been stored and archived.

### NotificationPage

A page used to view and read notifications, when this page is shown the “**Check notifications**” has been invoked. Events on this page will flow according to that use case.

#### QuestionsForYouList

This is a list of **Notification** entities that have been sent to the client regarding questions they may have the proper expertise to answer. This boundary will be responsible for rendering the appropriate type of notifications that correspond to these and handling transition.

#### AnswersForYouList

This is a list of **Notification** entities that have been sent to the client regarding questions that have been answered by an **Experienced User** actor in another use case. This boundary will be responsible for rendering the notifications and handling transition.

### SubmissionPage

This page is used to complete the “**Answer a question**” use case and “**Asking a question”** use case. Both require a form to enter information – and this page houses this form and the various controls. They are outlined below:

#### TitleTextBox

A simple textbox control that will map the **Question** *title* property with the user input.

#### DescriptionTextBox

A simple textbox control that will map the **Question** *body* property with the user input.

#### FormattingWidget

A widget that can control formatting options and mutate the *body* property accordingly; formatting details are not important at this stage.

#### SubmitButton

A button that when pressed can dismiss the **SubmissionPage** and return the results to the use case that required it.

### ViewTopicsPage

Generally, a page that will simply list questions which are generated from **Question** entities for the end user to view and visit.

### ViewQuestionPage

A page that is responsible for rendering a **Question** and by consequence, all of the associated **Answer** entities attached to it. When viewing this page, the “**Check a question**” use case has been invoked. These are done in the form of cards, described below:

#### SubmissionCard

Each contains information regarding the submission – body, date, score, and other attributes outlined on the according entity. Each card also contains a **Report** button which enables a **User** to generate an **AbuseReport** by invoking the “**Report abuse”** use case when touching it.

### ReportQuestionPage

A page that is responsible for handling the “**Reporting a question**” use case. It contains many fields pertaining to things like: reason for closing, description of the issue, and priority.

### ProfilePage

A page that is responsible for handling the “**Update a profile”** use case. This page will typically handle a stream of questions that allows updates to a **User** entity to be performed to have more pertinent **Notification** entities generated.

## Control Objects

In the following sections, the controls that are used throughout the sequence diagrams and use cases are outlined. Through the brief descriptions, you can get a glance at what each **control** object will perform. For a more detailed list of behaviors and exact interactions, one should consult the *Design* manual**.**

### Server Control

A basic controller that is responsible for delegating and interfacing with the asynchronous application server link. Most boundaries and other controllers that require sending information and fetching it will usually pass through this controller at some point to allow delegation of effort. The **ServerControl** is used to additionally request remote entities, such as **Question** and **Answer**.

### Authentication Control / OAuth Control

This controller has full responsibility for interfacing between the entities: **User** and **AuthorizationToken** and the **LoginPage.** Additionally**,** the controller provides services to communicate with the remote server that **Google Inc** has provided to authenticate the WLU *Google Apps* domain.

### NotificationControl

This is a controller typically used on the application server backend to do processing regarding where notifications about various entities and **Notification** entities should be forwarded. It also implements various services to push to the **Google Push Services** and ensure delivery to various users. Other responsibly include: forwarding messages, forwarding acknowledgements, and communication with the server control about request feedback.

### QuestionControl

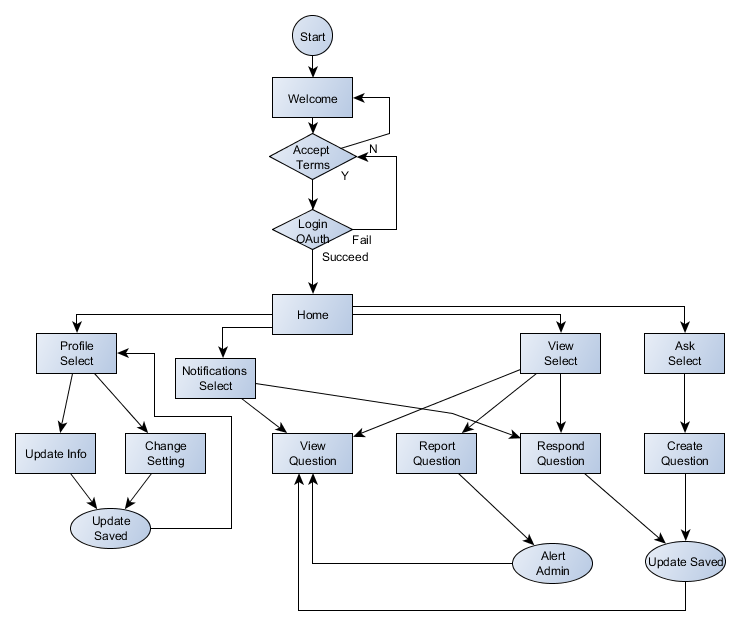
This is a control object with the specific duty of performing operations on controls, often used in various use cases such as “**View a question**” or “**Ask a question**”. Normally, the question control will operate on the local data and generated local entities. Remote entities can be requested by this controller by connecting and communicating with the **ServerControl.**

### AnswerControl

This is a control object with the specific duty of performing operations such as updating answers, submitting answers, on local **Answer** entities. The controller will often be used in use cases like “**Answer a question**” to communicate with the server. Normally, this control object will focus on operating on local entity data. However, remote entities are also requested by this control object by communicating with the **ServerControl.**

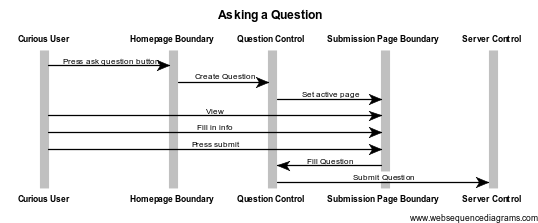
# Activity Diagram

An activity diagram is useful in showing the flow of use cases and how things can change throughout the application. A rough overview of these is provided below:

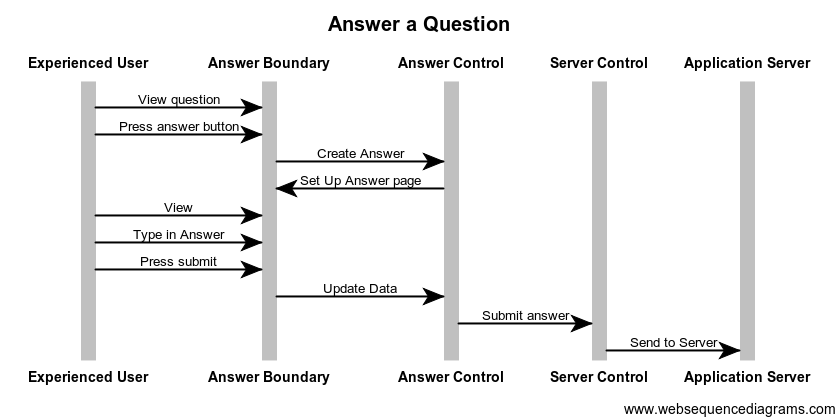


# Use Case Sequencing Diagrams

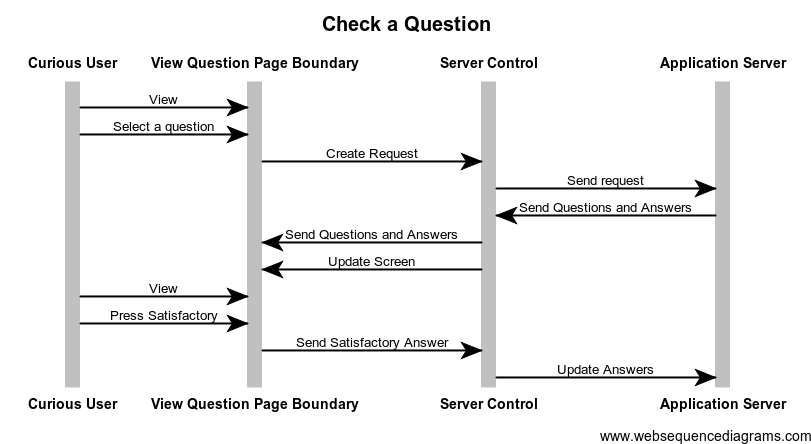
## Use Case Sequence: Asking a question



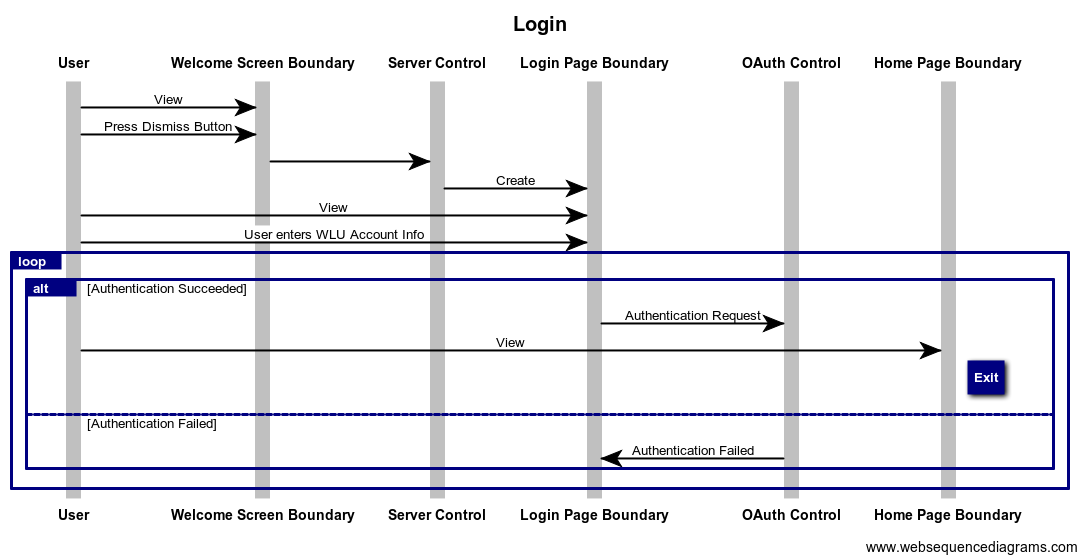
## Use Case Sequence: Answer a question



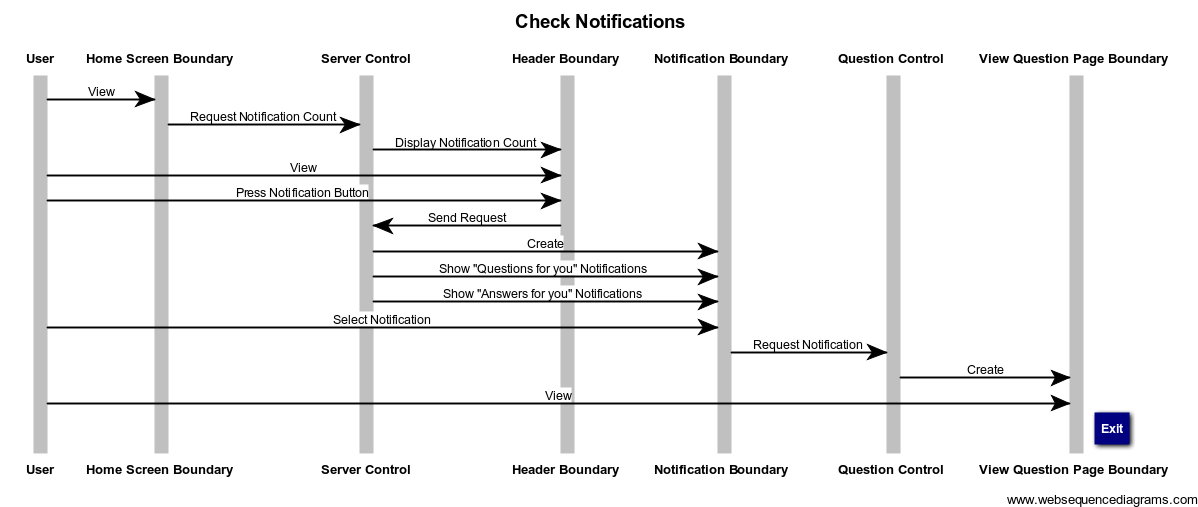
## Use Case Sequence: Check a question



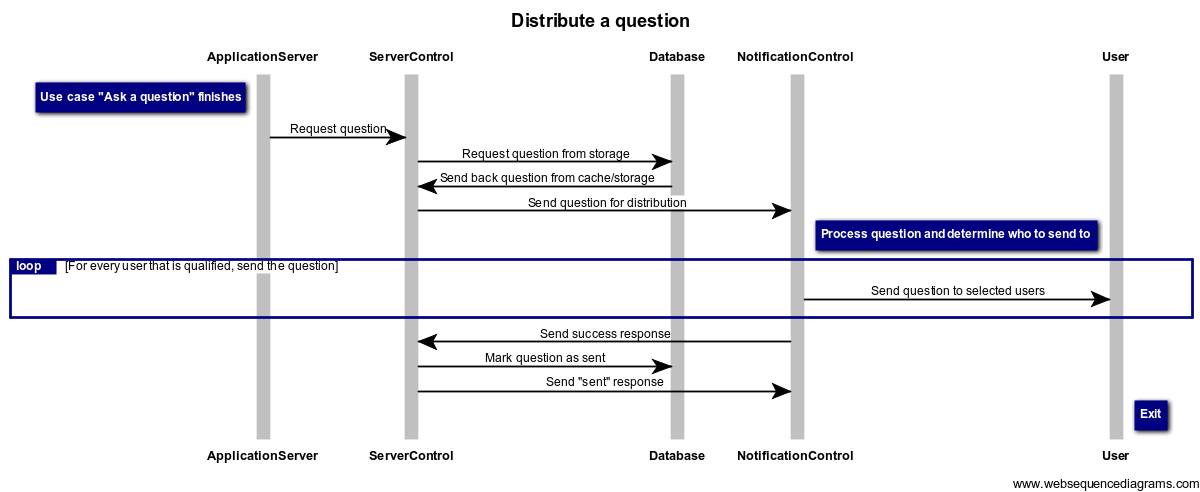
## Use Case Sequence: Login and setup

**

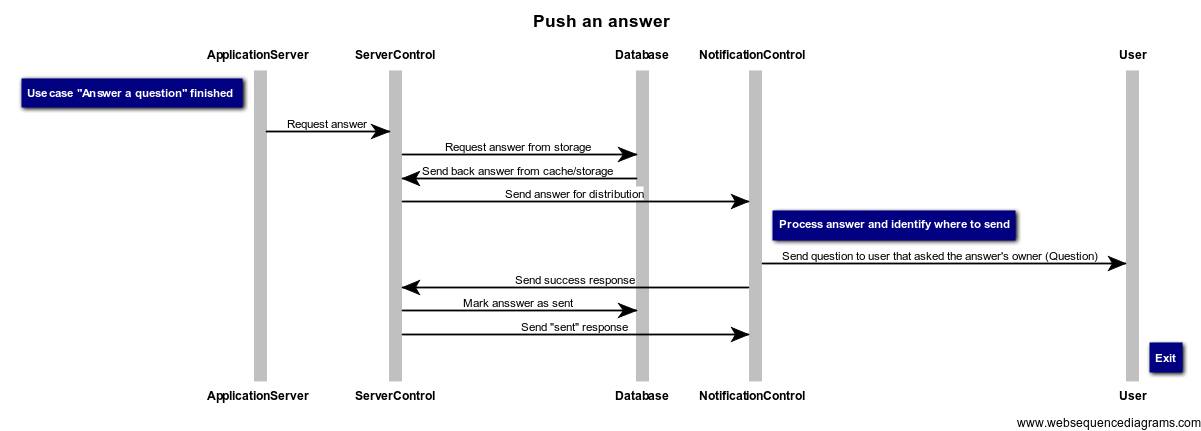
## Use Case Sequence: Check notifications



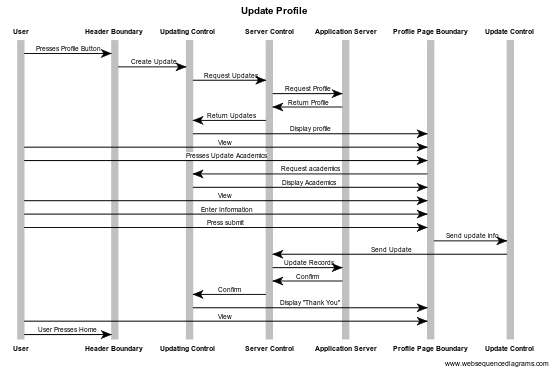
## Use Case Sequence: Distribute a question



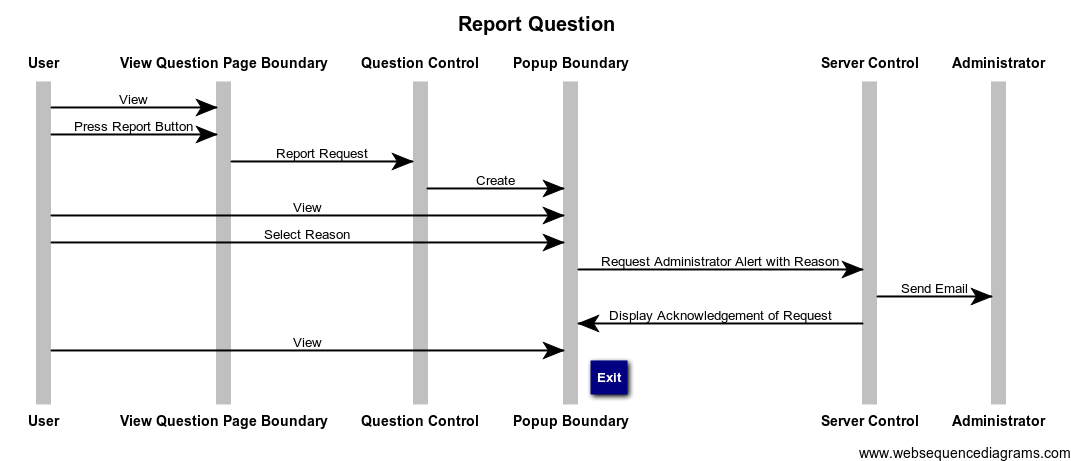
## Use Case Sequence: Push an answer



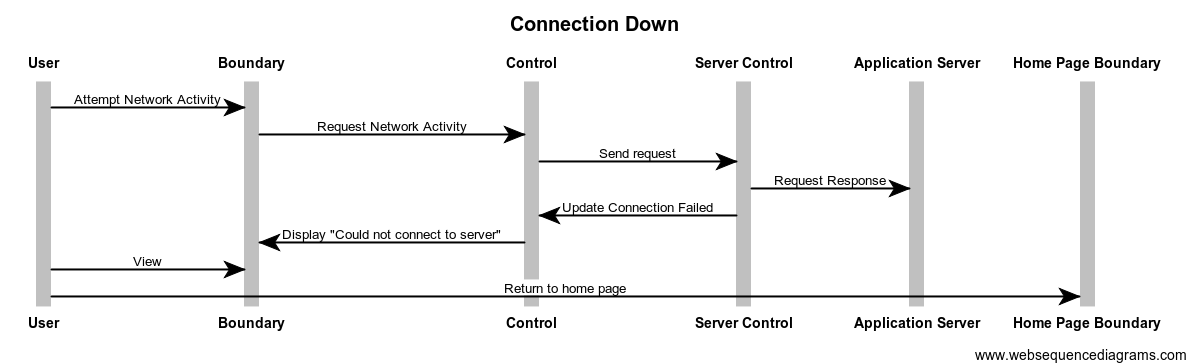
## Use Case Sequence: Update profile



## Use Case Sequence: Reporting a question



## Use Case Sequence: Connection Down



# State Diagrams

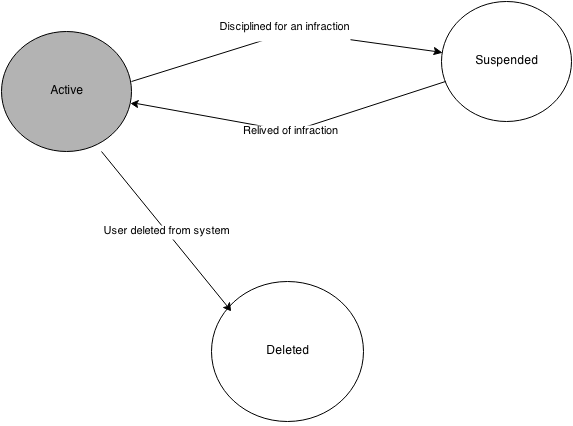
For many entities, a state diagram that summarizes potential states and transitions are described below. This list is as exhausted as possible, use the table of contents to navigate accordingly.

**Note:** The initial state for all entities is denoted by a solid gray state bubble on the below diagrams unless otherwise noted by a use case.

*For source files of the state diagrams, please check out the accompanied files in the complete report.*

## User

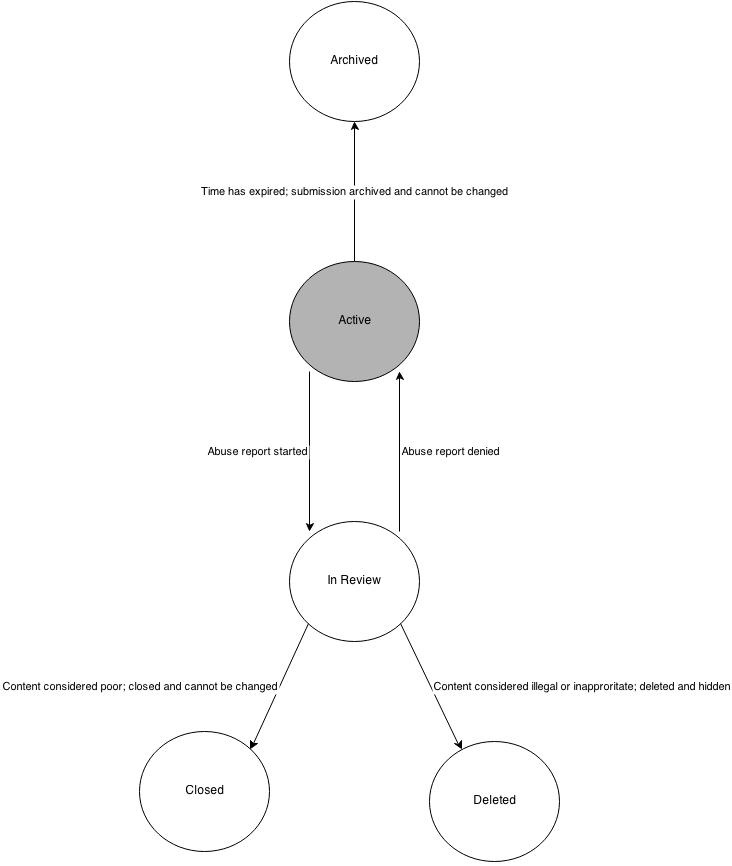
A **User** has a few states throughout the application that they can take on. Quickly, they are outlined here:



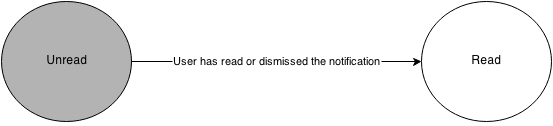
In this case, it is shown they can be **Suspended** or **Deleted** through administrative utilities.

## Submission

When a **Submission** is created, it is generated as **Active.** After some time, they can become archived. When invoking the “**Reporting a question**” use case, a **Question** entity may have their status changed to “**In Review**”.



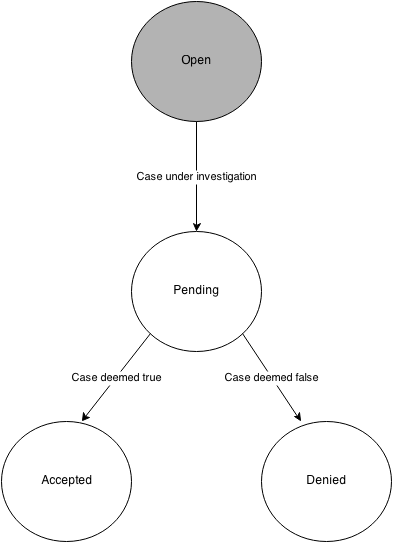
## Notification



The flow for this is very basic, however in the interest of being exhaustive it has been provided.

## AbuseReport

When a **AbuseReport** is generated, it is set to **Open.** As the report is handled, the entity is moved to **Pending** and then to another state when a decision has been made.



## AuthenticationToken

