CS673S16 Software Engineering
Team 3 - The Wishlist
Software Design Document

Team Member	Role(s)	Signature	Date
Duncan Morrissey	Project Leader, Configuration Leader	*Duncan Morrissey*	3/29/18
Edward Ryan	Design Leader, QA Leader	*Edward Ryan*	3/29/2018
Yiannis Karavas	Requirements Leader, Implementation Leader	*Yiannis Karavas*	3/29/2018
Zach Lister	Backup Project Leader, Management Plan Leader	*Zach Lister*	3/29/18
Ben Mitchell	Security Leader	*Benjamin Mitchell*	3/29/18

Revision history

Version	**Author**	**Date**	**Change**
2.0	**Edward**	**3/29/18**	**Updated diagrams for Iteration 2**
1.0	**Duncan, Edward, Yiannis, Zach, Ben**	**3/1/18**	**As issued**

1. Introduction

For our project we will be creating a gift registry website. The primary use is creating a shareable wishlist. Users will be able to create lists, add or remove items, and organize and rank these items within their list. Furthermore, they will be able to share these lists with others, and counterparties will be able to see what items have been purchased for an individual. The problem this solves is for events, parties or holiday times, we're creating an easily accessible place for people to share what gifts they want, and others can see what gifts have been purchased to help guide their buying behavior. The potential user is essentially anyone who takes place in a gift purchasing event like the holiday times.

The purpose of this document is to outline the software architecture of this application and to elaborate on the design patterns implemented and design decisions made.

2. Software Architecture

2.1 Ruby on Rails

The application is implemented using the Ruby on Rails web-application framework. Rails provides an object-oriented Model-View-Controller (MVC) base architecture with structures for database access, web services and web page rendering.

The Model framework relies on ActiveRecord, a Rails structure that combines the memory and database management into a single set of actions. This eliminates the need for a separate set of database provider classes or a large bank of previously defined stored procedures. Database schema is managed through ActiveRecord migration scripts which, using Ruby code, generate the described database tables. The migrations also make the database columns implicitly available as a part of a Model object instance, with additional behavior added based on the described relationships and validations in the Model itself.

The Controller framework contains a similar abstracted base class as did the Models: ActiveController. The controller is responsible for both providing information to the views to display to the user, as well as process user input into data that can be represented by the application models. The controllers are generally given a one-to-one relationship to the models (i.e. the User model would have a UsersController, the Group model a GroupsController, etc.).

The View framework uses a modified version HTML5, CSS, Javascript and Ruby to define the layout of the application and how the data is presented to the user. Working with the routing provided by Rails, the user interface functions as a single-page application (SPA). The

individual views interface with the appropriate controller (as determined by the router) based on information received from the user and retrieves/processes the necessary information to render the desired view. Views do need to be able to use the model structure to help render, but any changes to model instances are done through the appropriate controller.

2.2 Microsoft SQL Server

For the purposes of this application, ActiveRecord uses two configurations for database access: (1) a local Microsoft SQL Server 2017 instance for development and test environments, and (2) an Amazon Web Services (AWS) Relational Database Service (RDS) Microsoft SQL Server 2017 instance for the production environment. For the purposes of this application SQL Server 2017 Express provides sufficient functionality and scalability.

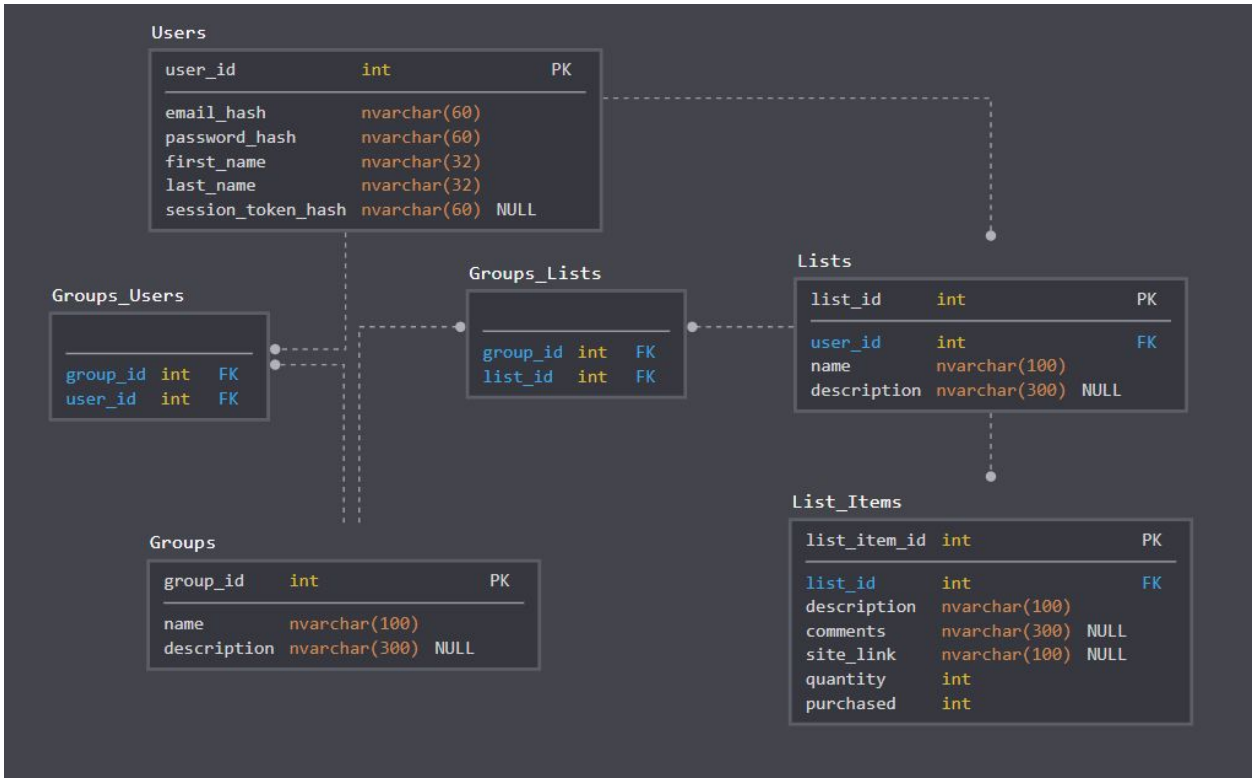

Figure 2.2 – The database schema for The Wish List application is shown.

3. Design Patterns

3.1 Model-View-Controller (MVC)

Rails serves as an implementation of the Model-View-Controller framework. In this framework, software is divided into three connected segments. The model serves as the manager of the data in the application, existing separately from whatever user interface is built on top. The View is the user interface itself, and controls how the user interacts with the application and its data. Finally, the controller is the liaison between the other two, tying user actions on the view to changes in the underlying data of the model.

Rails implements these abstractions directly, having directories for models, views and controllers.

3.2 Single Page Application

A single page application is the concept of using asynchronous requests to update data elements on a single web page without necessitating users to navigate and wait for separate pages of an application to load. It hearkens to the experience of local application experience on a desktop. While our application is not a single page application, it takes advantage of rails's support for asynchronous requests and routing, allowing lists and list items to be added and removed without page reloads.

4. Key Algorithms

4.1 Hashing (SHA-512)

Rather than handling user emails and passwords in plaintext, particularly in the database, hashing is used to allow for secure manipulation of the data. Hashing is preferred over traditional encryption, as it is a unidirectional operation - there is no "decryption" of the hash. This does come at the cost of not being able to reference the information elsewhere in the application, but that is currently not required.

5. Classes and Methods

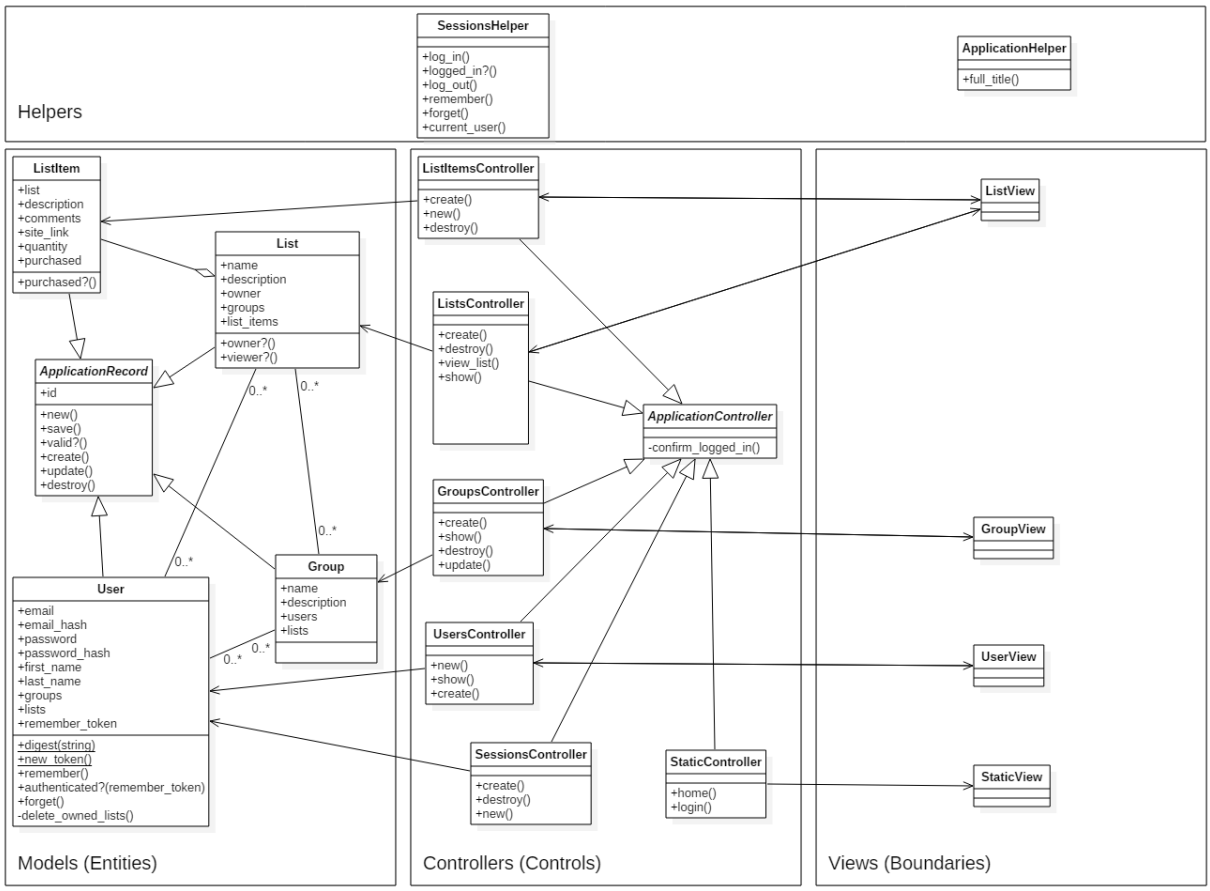

Figure 5.1 – The class diagram for The Wish List application is shown.

6. References

- Bootstrap (https://getbootstrap.com/)
- Bootstrap Sass 3.3.7 (https://github.com/twbs/bootstrap-sass)
- Ruby on Rails (http://rubyonrails.org/)
- Microsoft SQL Server 2017 (https://www.microsoft.com/en-us/sql-server/sql-server-2017)
- Amazon Web Services – Elastic Beanstalk (https://aws.amazon.com/elasticbeanstalk/)
- Amazon Web Services – Relational Database Service (https://aws.amazon.com/rds/)

7. Glossary