Software Design Document

February 16h 2017

Five Pixels

Sponsor:
Joy Knudsen

Faculty Mentor:
Dr. Mohamed Elwakil

Team Members:
Brandon Garling
Xiangzhi Cao
Matthew Nielsen
Mohammad Alsobhi
Clarissa Calderon

Rev 1.1

Introduction **2**

Implementation Overview **3**

Architectural Overview **4**

Module and Interface Descriptions **8**

User Module 8

Notification Module 8

Matching Module 9

Reporting Module 10

Customizability Module 11

Permissions Module 12

Setup Module 12

Implementation Plan **13**

Conclusion **15**

Introduction

The number of international students has been increasing in the United States by 72 percent since the year 2000. International students often have greater difficulty making friends than other students that are native to the United States; this affects international students in their academic performance. This may be in part because international students sometimes have a difficult time adjusting to American culture. Succeeding in a foreign country can be difficult, especially when you do not know anyone native to the country on a “friend” level basis. In addition, Americans often overlook the culture of foreign countries.

The program Flagfriends was originally started by Joan and Stan Alf in 2004. It was a non-profit organization called the Flagstaff International Friendship Program. The program aim was to help alleviate the transition of international students when traveling to the United States by pairing them up with local Flagstaff residents, commonly referred as hosts. Students and Hosts met up at least once a month and shared activities, such as: family dinners, picnics, holiday celebrations, sports activities, musical events, and other hobbies. Not only did international students gained helpful insight into the American traditions, hosts also gained insight into the student’s own past experiences and culture. Relationships between hosts and students sometimes outlasted the program and many hosts and students remained friends after the students graduated. In 2014, the program was passed to Joy Knudsen. The program which is now called Flagfriends is not a home-stay program, hosts and students establish relationships in the hopes that both parties will learn from one another.

Joy Knudsen is our sponsor and client for our NAU senior capstone experience. Joy has worked tirelessly to keep the program moving and to expand its potential. Currently the program serves around 80 international students and 80 hosts. The planned solution for Flagfriends is creating an online web app that will allow hosts and students to register and take part in the program. Hosts and students will be able to see each other and will be able to select each other. This planned solution will speed up the process of creating a “match” between a host and a student and to alleviate Joy’s job. Joy used to work on the host and student forms on her own manually reading all the forms and contacting hosts and students. With the web app, she will no longer have to take the time to form matches and will be able to focus her time on other tasks such as expanding Flagfriends by having other Universities use the program.

Flagfriends Users should access the site using HTTPS with a valid configured certification, the domain for the website is flagfriends.org and the site will use cookies and sessions to keep information between visits. When using this app, users will also be given the power to report other users who misuse the system. The point of this is to allow an administrator to ban users from using the system when appropriate. Users will be able to register for accounts after using the registration process, account data will be stored in a database and passwords will be encrypted prior to being stored in the database. Users will have a personal information form and a preferences form, they then will be able to match each other based on their preferences.

The point for having the website is to not only help Joy Knudsen but to also give hosts and students that wish to use the app the opportunity to easily sign up for this program in a quick manner. Users should be able to sign in for an account in 3 clicks from the homepage and the load speed for the pages shouldn't take longer than 2 seconds.

There will be a term of conditions that users will need to sign before using the system. The program will be explained and a FAQ page will be displayed along with contact information. The resulting system will be open source and will be free for anyone to use.

The purpose of this document is to explain the software design of the project for Flagfriends. We will talk about the different technologies that we use to develop the web app for Flagfriends. The process and approach we are taking to create the web app for Flagfriends is explored. The architecture and the user interface is explained in detail. User diagrams will also be explained in the following pages.

Implementation Overview

Our solution can be described as a general group friendship pairing web application we call Groupwise. This application will provide the client the functionality she needs to bring her organization online as well as provide a foundation for other organizations that want to run similar friendship programs. This application will be free, open source, relatively easy for new organizations to install and setup on their own infrastructure.

- Students and hosts will be able to create accounts, fill out an online application, and be in the system in a few minutes.
- Hosts will be able to view a list of all unpaired students in the application and send a friendship request to any students they may be interested in hosting.
- Students and hosts will be able to chat on the application in real time, before and after pairing.
- An administrator will be able to schedule check-up emails to be sent out to hosts and students to gather feedback and help resolve any possible issues that may arise.
- Hosts propose a match to a student, if the student accepts, they are committed to that host and vice versa.

In order to accomplish the technological objectives in this project we decided to employ a number of different technologies. To effectively talk about each piece, we'll split into two separate sections, one which describes our implementation overview for the frontend, and one which describes our implementation plan for the backend.

Our frontend implementation is based on an open-source front-end web application framework called AngularJS 2. AngularJS 2 allows us to easily make a single page application and gives a smooth user experience. It is largely based on a component architecture in which each component is responsible for a part of the application, this allows us to split our application into separate distinct pieces. We will also leverage a Javascript package known as Socket.IO which provides polyfills and a standard interface for WebSockets. Socket.IO gives us the ability to easily dispatch information from the backend to the frontend nearly instantaneously.

Our backend implementation is largely based around the open-source cross-platform Javascript runtime environment known as NodeJS. NodeJS lets us leverage pre-existing node packages that exist in an external repository known as Node Package Manager (NPM). These external packages provide functionality that we can leverage and combine into what will be the backend of our application.

Our backend implementation depends heavily on external node packages, they are: Express, Sequelize, NodeMailer, and Socket.IO. Express is the go-to when building web APIs in NodeJS, which is what the backend will end up exposing to the front-end. Sequelize is an Object Relational Model (ORM) which can be used to help with managing complex database objects and queries. NodeMailer gives us a simple interface to communicate with our external mail server and send email messages. Again, Socket.IO is needed on the backend to have something for the front-end Socket.IO to communicate with.

Finally, we utilize MariaDB to store our data. MariaDB is a MySQL implementation that is optimized for smaller installations and ease of use, that makes it perfect for our use case. We plan to support additional database types on our application which will be configurable during the setup process.

While developing Groupwise we will be using JetBrains's WebStorm IDE. WebStorm gives us the most flexibility and compatibility with our given software suite. We also will be using Travis CI for continuous integration testing when we get to our testing phase.

Architectural Overview

Our application takes on a more 3 tier architecture style, with the separation of the presentation, business logic, and data layers. The application will employ a number of communication methods in order to allow communication between these different tiers. These include REST communication, database communication, SMTP communication, WebSocket communication, and SQL communication. Much of our underlying business logic is split into separate components (or modules), these include: user module, notification module, matching module, setup module, permissions module, customizability module, and reporting module.

Each underlying business logic module will hook into one or more communication method in order to achieve its purpose. Many of the modules will interact with each other through publicly available methods. By allowing cross-module communication we can allow for more flexibility in our architecture.

Each module has a compliment on the Angular front-end side of the application as well that will allow for the visual representation of the module. In this case, we leverage AngularJS 2's built in constructs to componentize our application. In this way, we can provide components that are rendered, services that communicate with the backend, and models which help with data transfer and consistency between the business logic and presentation tiers.

Figure 1: Physical system architecture

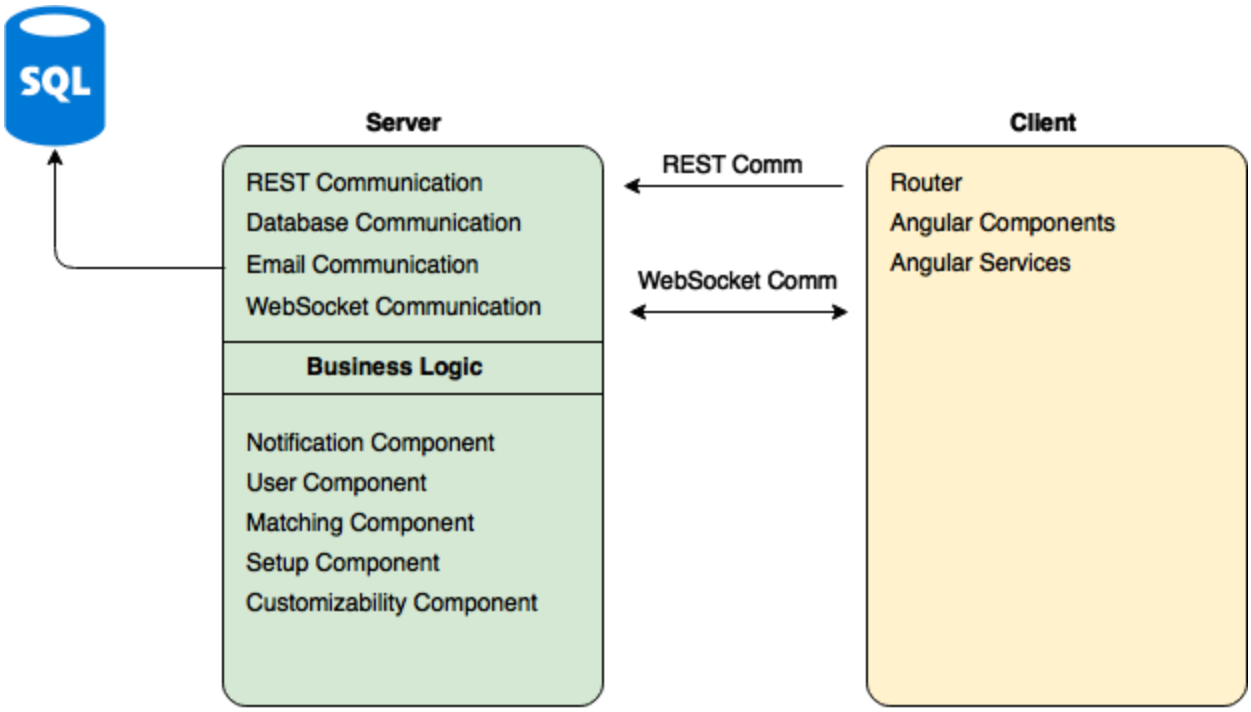
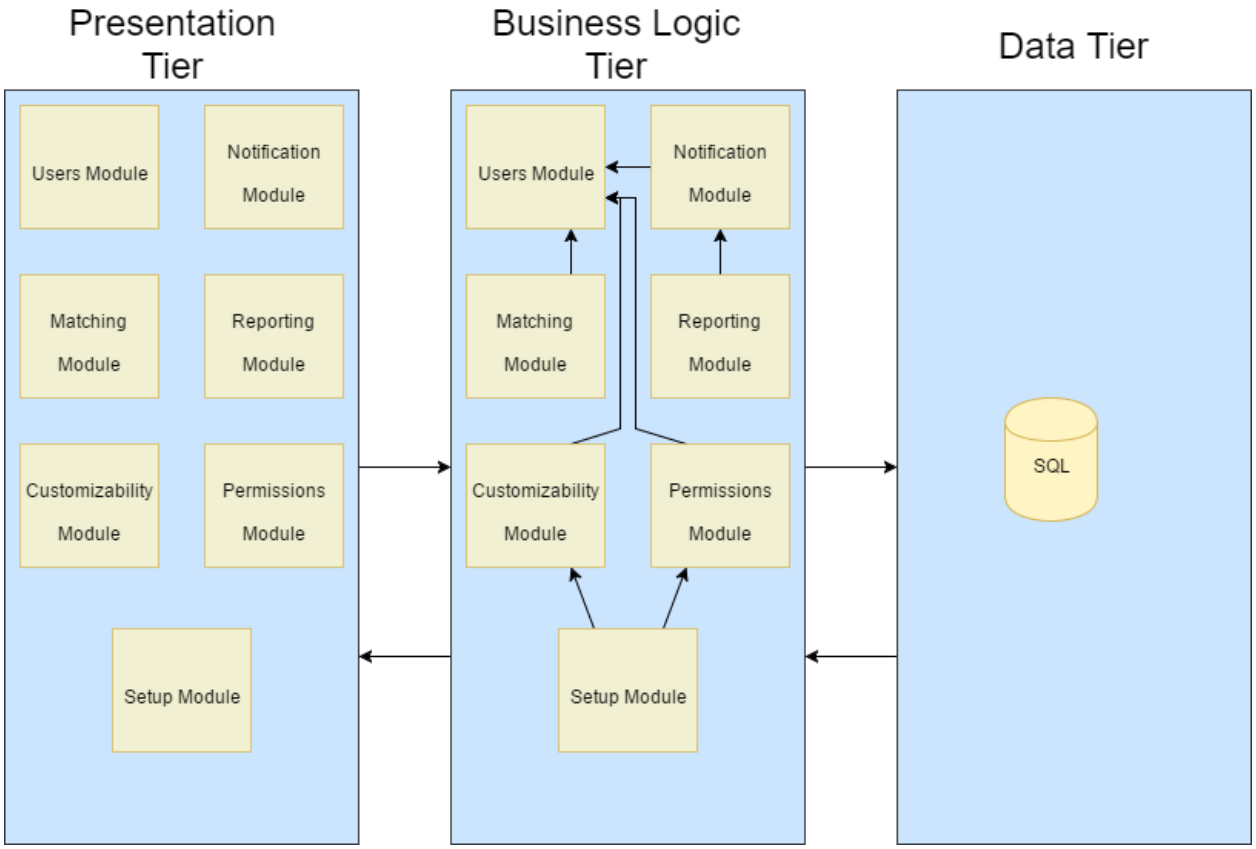


Figure 1 above shows the physical architecture of our system. The server ends up handling all authoritative business logic while the client is used to present the underlying data and call services on the server which in turn edit underlying model data. By leveraging a number of communicators, we can achieve all underlying requirements of the system. One can also begin to see the underlying 3 tier architecture beginning to show.

Figure 2: Software architecture



As can be seen above in Figure 2, our architecture closely represents a classical 3 tier architecture. Each module has a presentation component and a business logic component. By using this pattern, we can separate out the functionality of each module and define primary responsibilities for each one.

Figure 3: System-wide UML class diagram

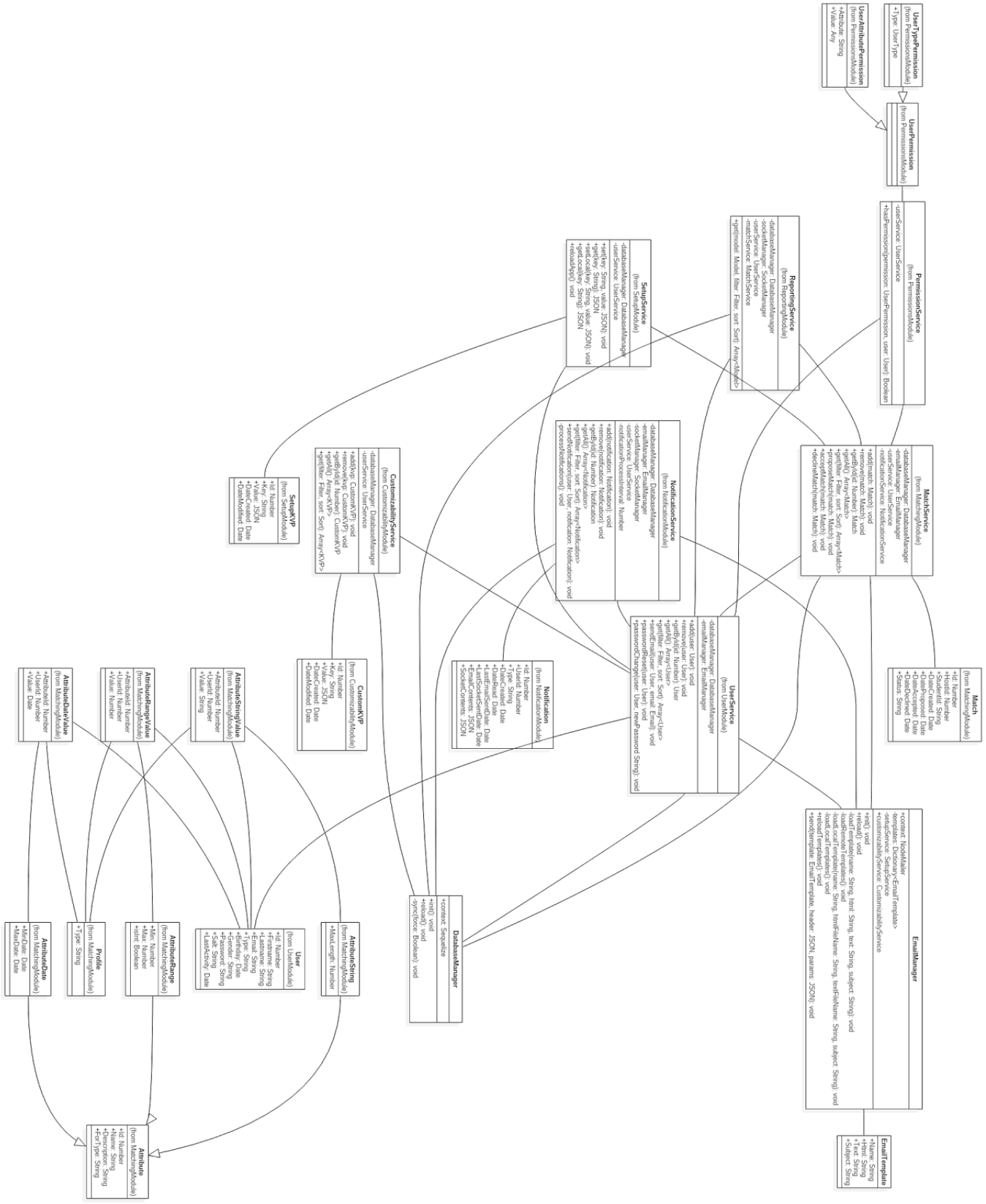


Figure 3 in the previous page shows our business logic layer system-wide UML class diagram. This figure shows how every single piece connects with each other. We will further analyze each piece in the next section below.

Module and Interface Descriptions

User Module

Groupwise will support three distinct variants of users, hosts, students, and administrators. The user module is responsible for the creation and management of these different users. It will provide an interface for other modules to communicate with users in the system. This module will leverage the Sequelize ORM package in order to keep users persistent in the application. Most of the low-level business logic pertaining directly to users will live in this module. This includes actions and handling information such as: registration, password reset, account management, personal settings.

Figure 4: User Module UML diagram

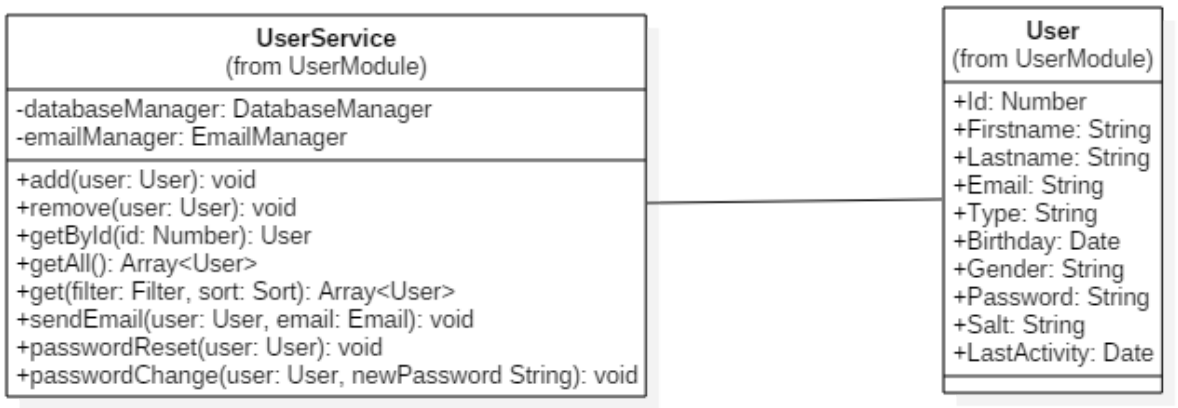


Figure 4 above shows a service singleton UserService which helps to manipulate users within the application. It provides useful functions such as passwordReset and passwordChange which help to handle complex operations regarding setting passwords.

Notification Module

The notification module is a simple module that interfaces with the User module and will provide the ability to schedule jobs that will run periodically throughout the lifetime of the server application. It will be able to do custom queries on the users and determine which users should receive which notifications. It will also provide a simple interface for other modules to send

simple notification messages. This module will be able to provide both immediate front-end notifications via Socket.IO and email communication via NodeMailer.

Figure 5: Notification Module UML diagram

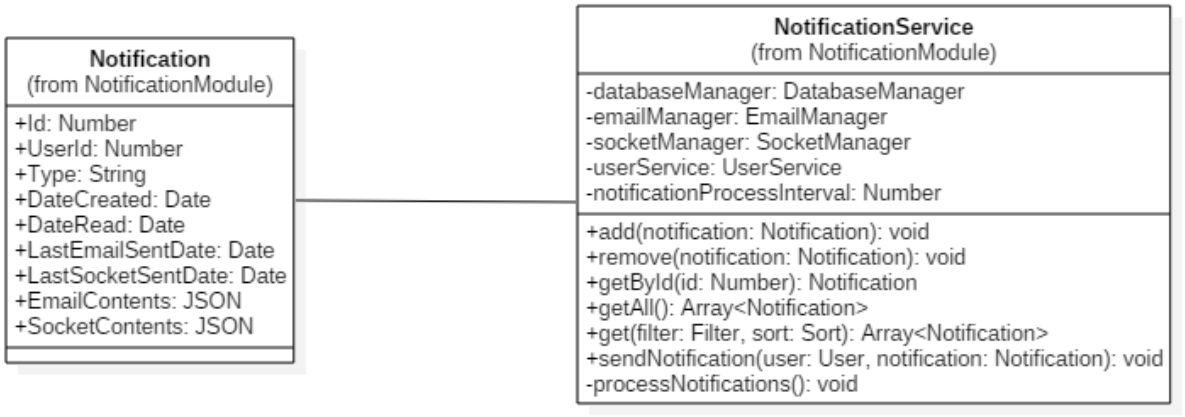


Figure 5 shows a service singleton NotificationService which helps to manipulate notifications within the application. This service will handle synchronizing all notifications with the database as well as dispatching notifications via Socket.IO as well as through the Email service. A Notification will be stored in the database and can be retrieved by the NotificationService during a specified configurable interval. This allows the Groupwise application to be restarted without losing pending Notifications that need to be sent or resent to users.

Matching Module

The matching module will be one of the simpler modules in the system. It will handle most of the underlying business logic relating to matches, and dispatching notifications via the notification module.

Figure 6: Matching Module UML diagram

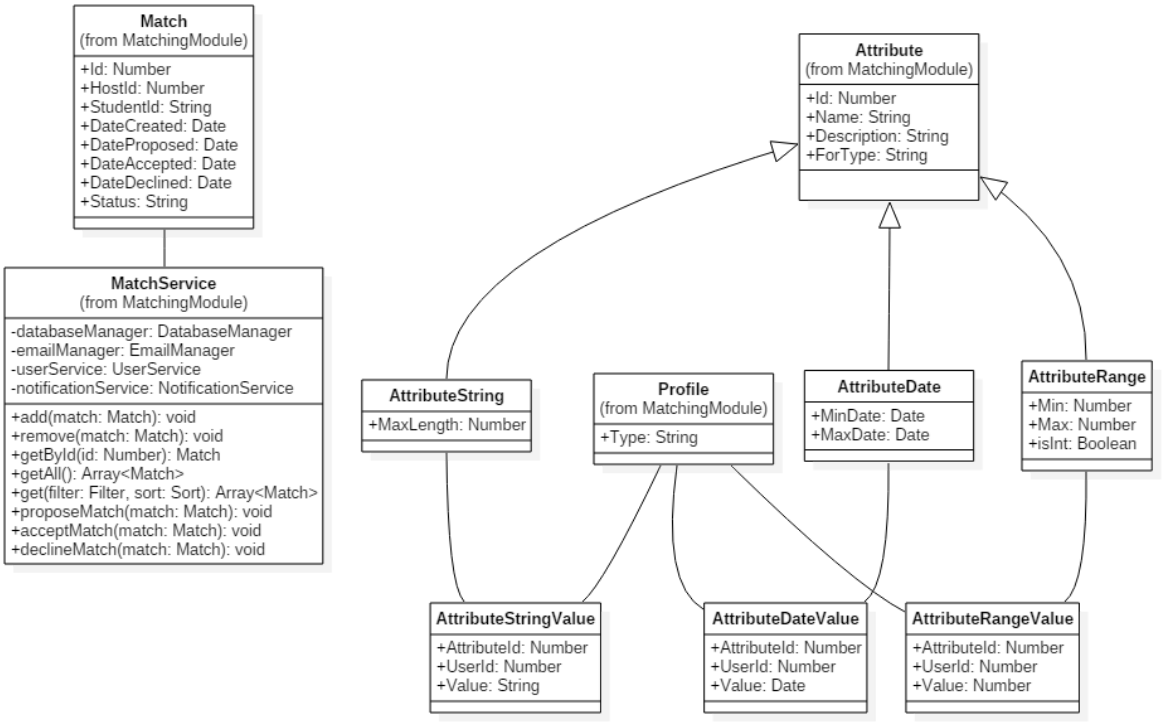
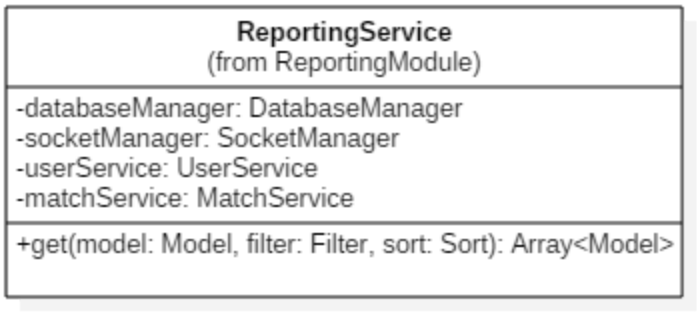


Figure 6 shows a MatchService singleton which can be used to manage matches between users in the system. A match can be managed directly through this MatchService, and the appropriate actions will be performed on a proposeMatch, acceptMatch, or declineMatch call. In addition, we show a bit of how Attributes will work in the system. An administrator can add and configure a number of different Attributes which can be attached to either a Host or a Student user type, this allows for a great level of customizability in the application.

Reporting Module

The reporting module will essentially be a wrapper around different database objects, the front-end interface will handle what to do with all the collected information about all the database objects.

Figure 7: Reporting Module UML diagram



The reporting UML diagram shown above in Figure 7 is simple, it gets data from the database based on the type of model requested, the filter function provided, and sorts the results based on the sort function provided. With this we can add a REST API endpoint in which we can specify all these options and retrieve a list of any type of entity we wish to expose to the administrator for auditing and reporting. Then the presentation layer can handle presenting this information in a meaningful way.

Customizability Module

The customizability module will handle any high-level site customizability. Administrators will be able to change customizability options and they should be able to edit the website name, landing page, and FAQ page.

Figure 8: Customizability Module UML diagram

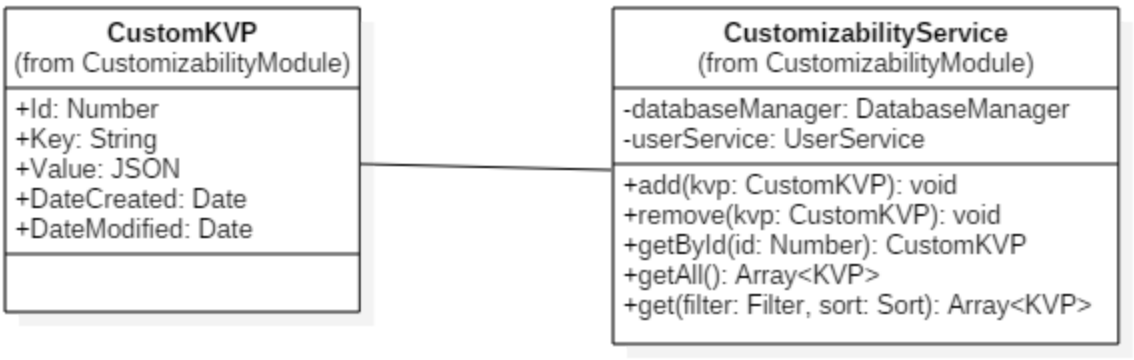
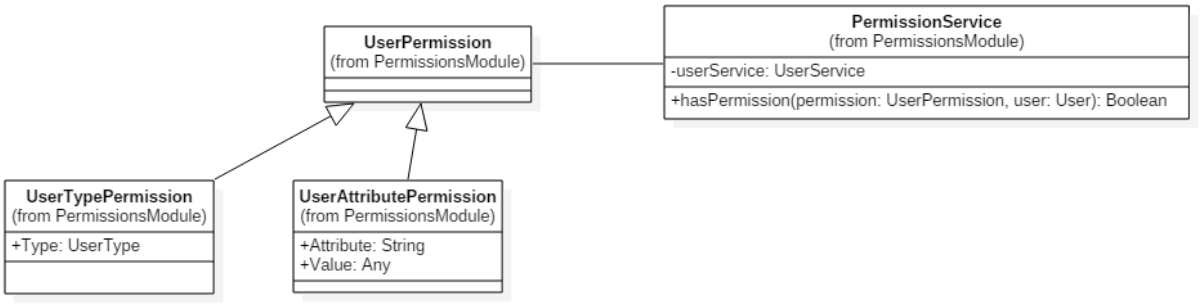


Figure 8 shows the Customizability Service class which simply handles the storage and retrieval of KVP objects. This allows the greatest level of flexibility, this allows us to store whatever information we may want to store in the KVP, whether it be HTML, text, or some other value.

Permissions Module

There will be a permissions module for the backend, and many permission guards on the front-end. This is to ensure that each user type has access only to the information that they need to know about, and nothing more.

Figure 9: Permissions Module UML diagram

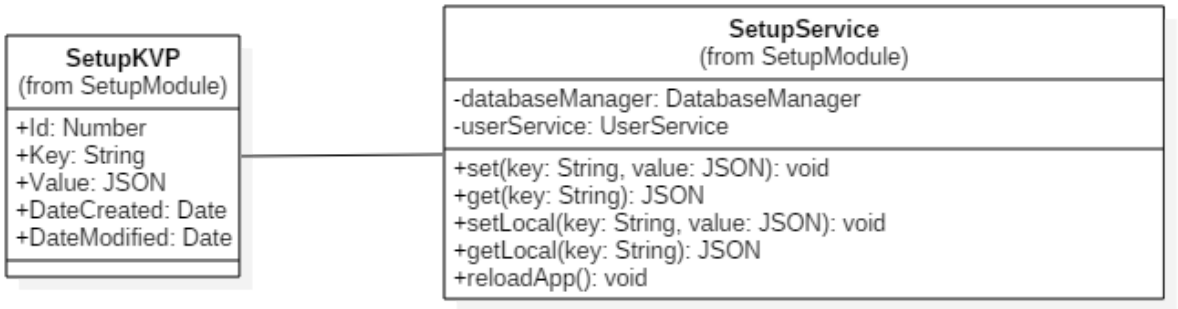


The permissions module shown above exposes a PermissionService which can be used to check if a specified user has a specific permission specified by a UserPermission.

Setup Module

The setup module will allow fresh installations and existing installations of the application to configure settings such as database connection info, mail account information, and general configuration information such as the site root URL.

Figure 10: Setup Module UML diagram



The setup module shown in Figure 10 is similar to the customizability module in that it stores KVPs. The main difference is that this store of KVPs is much more low-level systems operation level important. We also provide a getLocal and setLocal method which allows reading and writing to a local static configuration file that may contain connection information such as

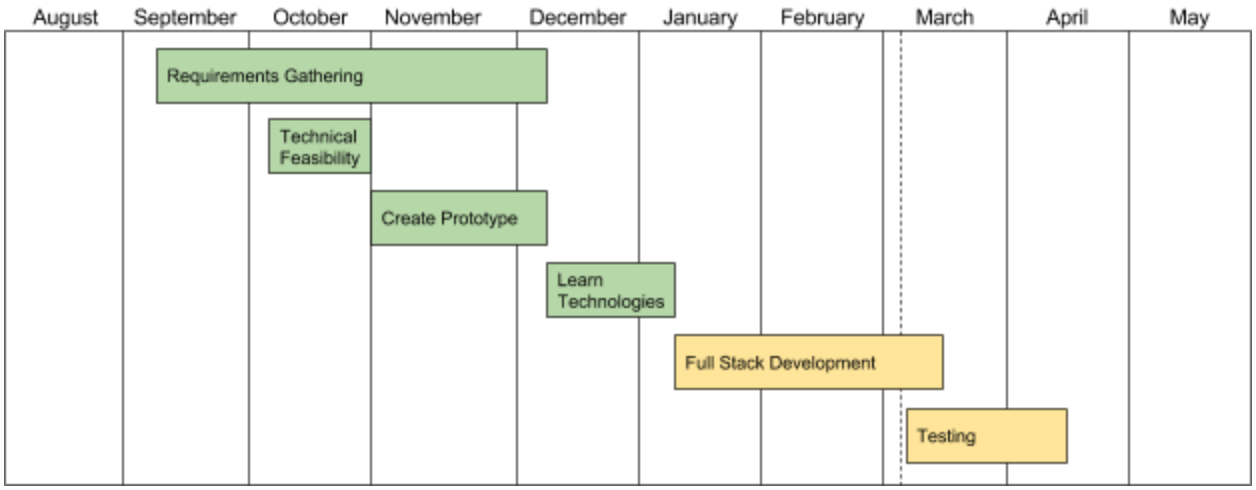
database connection strings, settings needed to even access other data. Providing a reloadApp() function we can quickly reload the application and apply any changed settings without having to restart the server application.

Implementation Plan

To ensure that the project is completed on time. We are using Agile/Scrum methodology, breaking our project into sprints. These sprints together make several milestones that help to track the progress of the project.
Upcoming milestones:

- Matching functionality
- Reporting functionality
- Customizability functionality
- User testing
- Acceptance testing

Figure 11: Implementation Plan



Our project plan is to divide our tasks, as can be seen above in Figure 11. We plan to be working on notifications in parallel, and having that done by early February. Full matching support will be implemented by mid-February, quickly followed by reporting support. Customizability will be worked on in parallel with reporting support, we foresee this as being one of the longer milestones in the project. Finally, we plan to begin user testing in March, while working on fixing bugs and any other final changes to the project. Acceptance testing will be performed at the end of March into April. By April the project should be completed and ready to be used.

Table 1: Division of work

1/17 - 1/21	Web app front end: Mohammad, Clarissa	Web app backend: Brandon, Matt, Xiangzhi
1/22 - 1/28	Web app front end: Mohammad, Clarissa	Web app backend: Brandon, Matt, Xiangzhi
1/29 - 2/4	Design doc draft: Mohammad, Clarissa, Xiangzhi	Web app backend: Brandon, Matt
2/5 - 2/11	Design doc draft: Mohammad, Clarissa, Xiangzhi, Brandon, Matt Small web app tasks: Mohammad, Clarissa, Xiangzhi	Web app backend: Brandon, Matt
2/12 - 2/18	Final design doc: Mohammad, Clarissa, Brandon, Xiangzhi, Matt Web app tasks: Mohammad, Clarissa, Brandon, Xiangzhi, Matt	Web app backend: Mohammad, Clarissa, Brandon, Xiangzhi, Matt
2/19 - 2/25	UGRADS registration: Web app tasks:	Web app backend:
2/26 - 3/4	Web app tasks:	Web app backend:
3/5 - 3/11	Design review presentation: Web app tasks:	Web app backend:
3/19 - 3/25	Software testing plan: Web app tasks:	Web app backend:
3/26 - 4/1	Design review presentation: Web app tasks:	Web app backend:
4/2 - 4/8	Capstone poster: Web app tasks:	Web app backend:
4/9 - 4/15	Team website: Web app tasks:	Web app backend:
4/16 - 4/22	Capstone conference: Web app tasks:	Web app backend:
4/23 - 4/29	Acceptance test demo:	Web app backend:
4/30 - 5/6	Web app backend:	

To ensure we get these tasks completed, we have employed a few techniques. These include the use of Trello, a scheduling application in the form of tasks belonging to certain “boards”, more specifically “backlog”, “in progress”, and “complete”. We also use a Slack chat application to boost team communication. Specific rules for these applications are also specified. In Trello, due dates are set and should be adhered to. In Slack if a team member posts something that requires a response the response should be given in a timely manner, specifically within the same day or the morning of the next. Team communication is imperative in the timely completion of this project.

Conclusion

The team 5 Pixels is grateful to have received the opportunity to work on the program Flagfriends. We are doing our best to help Joy Knudsen bring forth her vision of creating a web app to ease the process of signing up to be considered a host or student in the program. This program will help international students get the help they need to succeed and will also allow Joy promote and expand the program with marketing and social activities.

By having this program be a simple process of users signing up and matching themselves we can allow this software to benefit other organizations that wish to use the app. Organizations that wish to help international students will be able to pick this up and have a working platform up and running.

5 Pixels has proved that we have the skills to work and complete the project. We look forward to completing this project and to help international students obtain the help they need.