BUDDY SCHEME

SEG Major Group Project

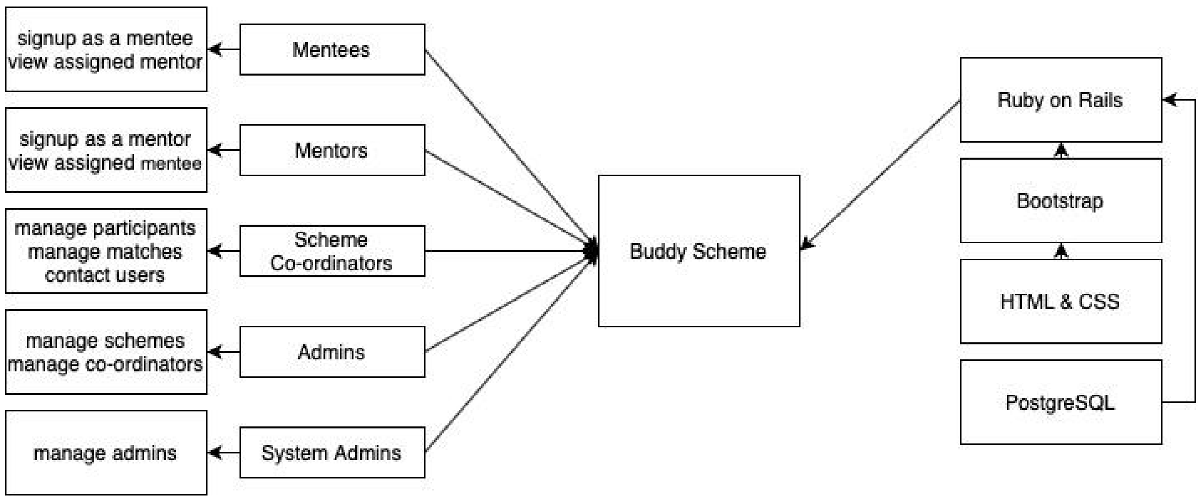
Client
Jeroen Keppens
Department of Informatics
King's College London

Authors
Christian Impollonia - K1925245
Ork Hossain Muntaqin - K19016467
Daniel Van Cuylenburg - K19012373
Mohammad Rahman - K2036243
Marouane El Moubarik Alaoui -K1897234
Suleyman Ahmed -K19036135

CLIENT'S OBJECTIVE

The buddy scheme is a system designed to match new students with senior students, and in informing participants of the matching process's outcome. This has been accommodated to support signing up even without a King's account. The application matches mentors and mentees, announces pairings, and allows the creation and deletion of buddy schemes. This is also done by allowing mentees to impose constraints on their mentors, and vica versa, through requesting that they be assigned a buddy of the same gender. Additionally, the buddy scheme aims to pair mentors and mentees based on shared interests. This is all done by asking participants to provide relevant information during signup, and by giving scheme coordinators the ability to reuse existing forms, share information with scheme participants, and the ability to contact all potential participants without access to a mailing list.

SYSTEM ARCHITECTURE



The block diagram above shows the architecture of the Buddy Scheme. It has five types of users and all of their actions are described on their left. To the right of the diagram, the technology stack is shown. It includes HTML and CSS for the design and construction of forms and web pages, PostgreSQL for the database and Ruby on Rails for the framework. More information on the technology stack can be found in the next section.

TECHNOLOGY STACK USED

We used:

- Ruby on Rails
- Bootstrap
- Devise
- Html and Css

For the buddy scheme, we decided to use Ruby on Rails as the framework to implement this full stack application. We also used PostgreSQL to generate our database, and the ruby gem devise to handle user authentication. Ruby on Rails made the most sense for us to use as this is the framework we were all most familiar with, due to using it for a previous project. This framework provided us with structure for a database, webpages, and localhost for web services. It also made it easier to incorporate HTML and CSS, all while having some embedded Ruby that made it simpler to build the app. Ruby on Rails also makes it easier to run automated tests, which this report goes into further detail in the Quality Assurance Section.

QUALITY ASSURANCE

Several approaches were used in testing and inspecting our Buddy Scheme. Ruby on Rails makes it easier to write and run automated tests. Ruby on Rails produces skeleton test code that mimics the models and controllers created during development. This is done automatically by having a test directory that holds different tests used in different directories in development. We used the test directory to hold most of our tests. Additionally, we have also added several gems that aid in facilitating automated testing in Ruby on Rails. An example of this is Rails Minitest. When running the command “rails test “, automated testing, similar to Gradle, is used by rails without the necessity of writing a build script. We installed rails mintiest as it facilitates automated testing and works as a test reporter. After running the tests, the results are formatted in a more concise way that helps us see if there is any failed assertion.

We have implemented additional testing for the controllers. Primarily, we focused on testing the user controller and having several assertions that redirect the user when they update their accounts, or deletes a user when one chooses to delete their account.

For the user tests, we have also implemented model testing in the user_test.rb file that makes assertions to check whether or not a user is valid when signing up. These include testing to see that different passwords should not be valid, and also making sure a user is only valid with the correct parameters. Users with incorrect parameters such as an invalid

email address or role would have the ***assert_not user.valid?*** , making sure that the test yielded an invalid user result.
To test the matching algorithm, we created 5 mentors and 5 mentees in the *users.yml* fixtures file, and then created 5 separate tests. Each test simulated the matching algorithm as if the scheme coordinator had selected one of the four user's descriptive fields to match by. The test asserts that a match has been created between two users that shared the same ethnicity, religion, gender or second language respective to each test. The fifth test asserts that a match has been created between two users that have selected to be matched by their gender constraint.
We also created an alternative main branch, called dev, where we would push the code before officially merging it with the master branch.
Before merging a pull request to the dev branch, two members of the group are required to approve it. This is done so that two members of the group can test and run any changes made on the separate branch, ensuring that everything works as intended, and that any merge conflicts have been correctly resolved.

PROJECT MANAGEMENT

For this project, the team chose to use an agile approach. Meetings were held twice a week, Tuesday's and Friday's, where we discussed any problems that had arisen since the last meeting, talked about features that needed to be implemented, and assigned new tasks. We set up a Trello board with various use cases for our project and assigned different colours to each use case to represent which area of the project they belonged to. Team members either chose or were assigned tasks based on their skill sets, and were responsible for completing each task in a set amount of time, usually in one week sprints, as per the AGILE method. Code was shared through the version control system GitHub, where we set up a repository and created new branches for every task assigned. Once the code on the individual branches was complete and worked correctly, the author would submit a pull request to merge their branch with the main code. This pull request had to be checked and approved by two other members of the team to make sure only the best quality code made it to the final product. Other than Trello and GitHub, the team used WhatsApp to communicate any issues they were facing and to coordinate meetings.

TEAM ORGANIZATION AND INDIVIDUAL RESPONSIBILITIES

Christian:

He mostly did coding, as seen on Trello. He implemented user authentication and handled general database interaction for most of the user roles. Outside of coding, he made the design overview on Figma, he wrote the README, and handled app deployment on Heroku.

Ork:

He had roles both in the front end and the back end of the program, as for seen from the Trello board. He focused on the cleansing of code, enhancing the visual appearance,some part of the creation of the program, writing the developer instructions and the styling of the report.

Daniel:

Most of the tasks Daniel undertook can be seen on the trello board. He was mainly involved in writing and editing functions for controllers and views, such as the matching algorithm. Daniel also worked with and edited the database, wrote tests, developed the emailing system, fixed various bugs throughout the project, and edited the report.

Mohammad:

Designed the application UI, and integrated this throughout the entirety of the application using ruby and HTML, ensuring that all pages are routed correctly. Furthermore, created the final screencast demonstration of our project, and was involved in various back end bug fixes along the way.

Marouane:

His main responsibilities varied throughout this project, though he was mainly in charge of implementing quality assurance and testing for the buddy scheme. He was also pivotal in writing the report and working with his teammates whenever it was needed.

Suleyman:

His responsibilities throughout the project were very diverse. He was part of the back end team and played an important role in developing the controllers and the views. Suleyman also helped in making the routes and with the testing. He was also an important part of writing the report.