Computer Science
UNIVERSITY OF TORONTO

CSC309 Team 02: CollegeChef

A web app to find recipes with only the ingredients you have.

Team Member 1 – Xiaohan (Becky) Zhou	Email: xiaohan.zhou@mail.utoronto.ca
Team Member 2 – Huimin (Cinny) Cao	Email: cinny.cao@mail.utoronto.ca
Team Member 3 – Sean Carroll	Email: morgan.carroll@mail.utoronto.ca
Team Member 4 – Tanay Onder	Email: tanay.onder@mail.utoronto.ca

1 Description & Requirements

College Chef is an online recipe catalogue where the user can search for recipes by the ingredients instead of only the name of the recipe. Our website is tailored for university students who don't have the luxury to choose what they want to make for dinner, the website allows them to use what they currently have in their fridge and make the best of it. At the beginning of the project we identified 104 requirements, 70 of which we deemed achievable.

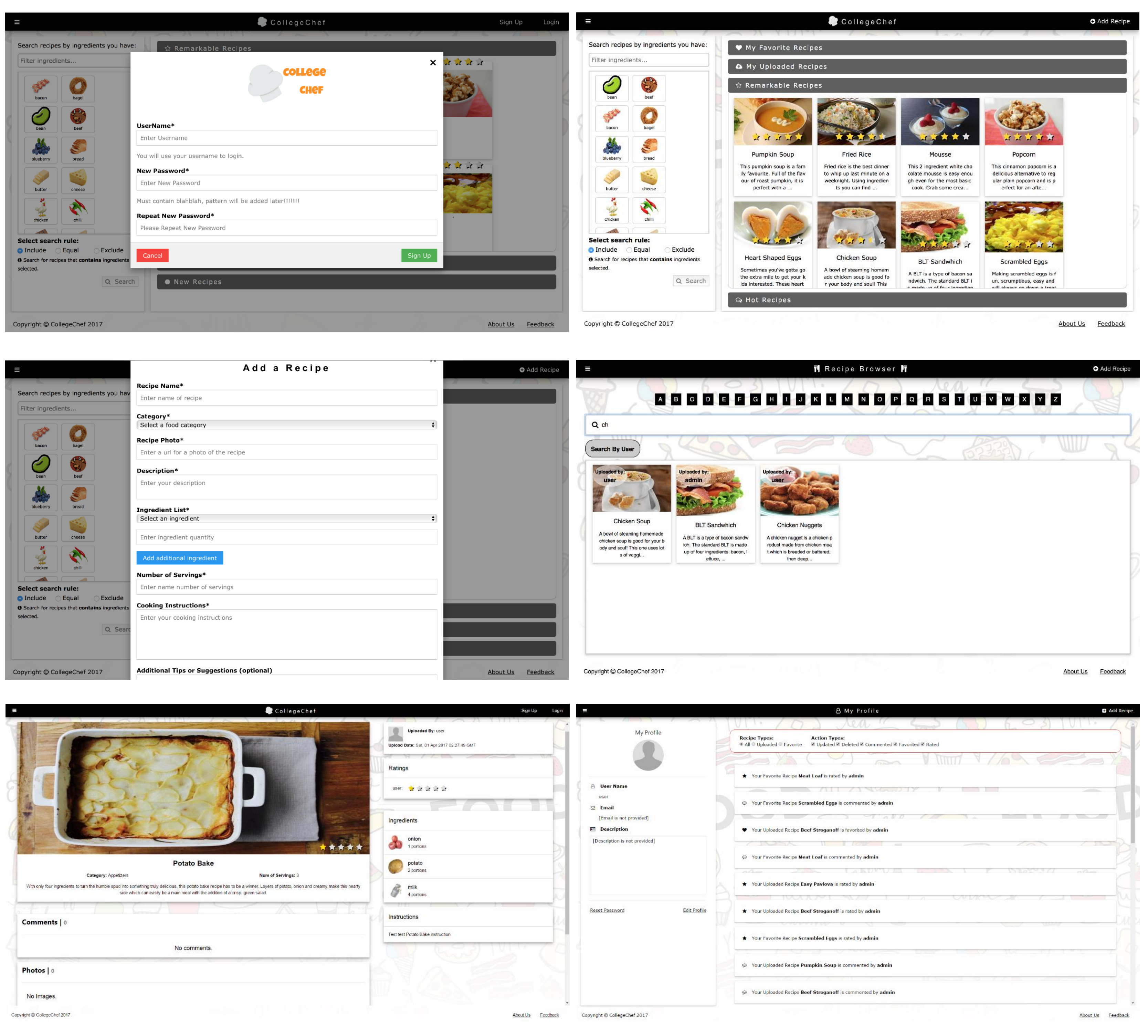
Some of the most important requirements were:

- Include/Exclude/Only search functionality
- Different views for favourited and uploaded recipes
- User signup/login capabilities

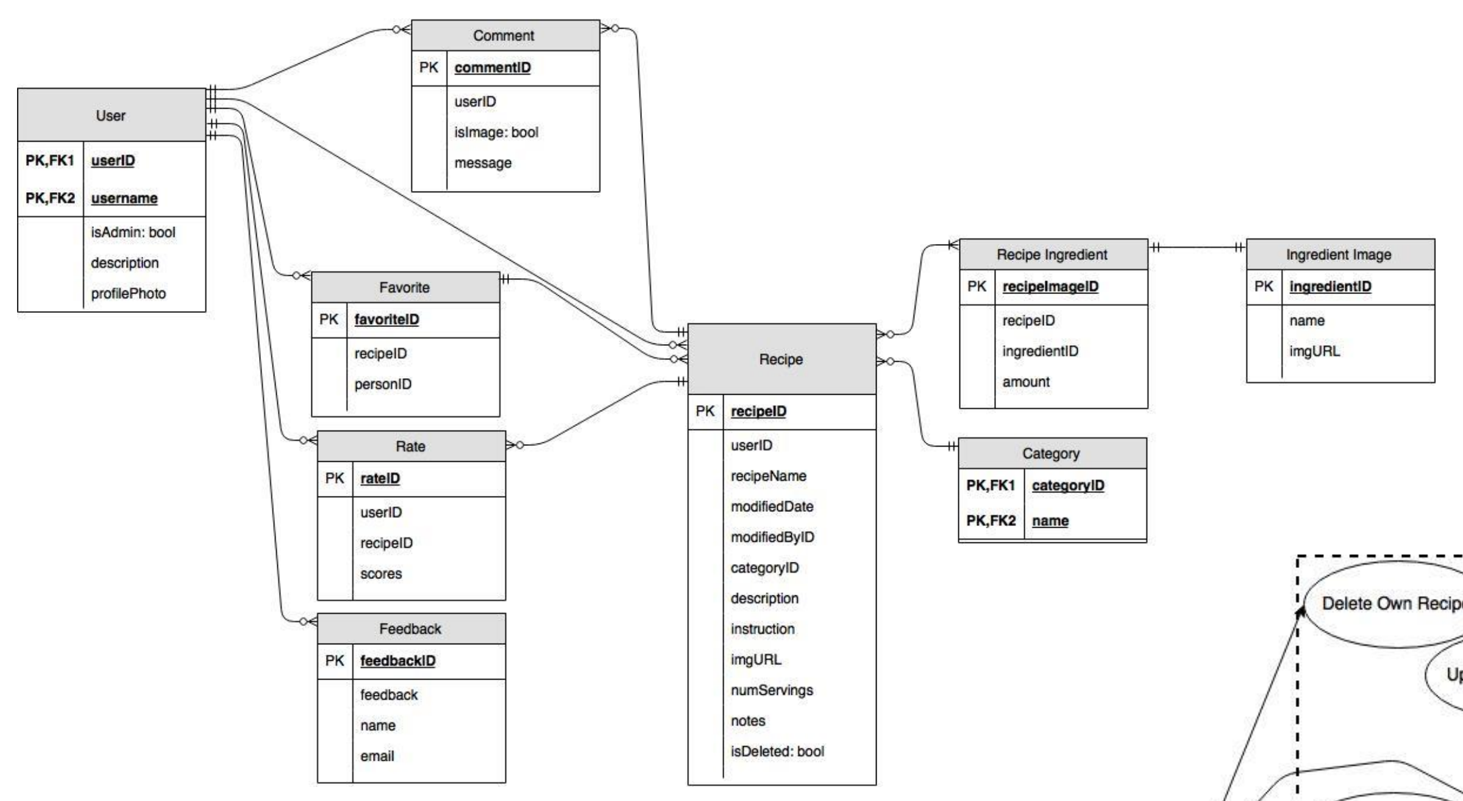
2 Features

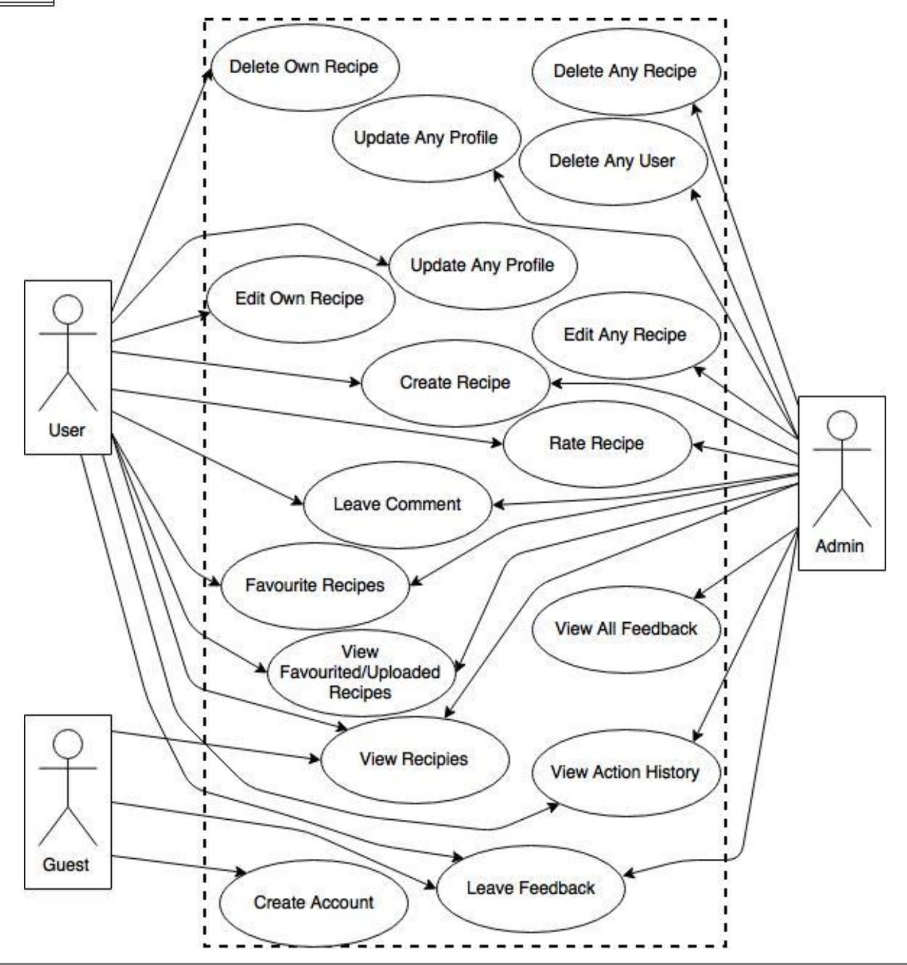
- Client-server model, communicating by way of HTTP
- Responsive design
  - Presentation is adjusted for mobile and screens of different sizes
- Semantic HTML such that the appearance of our webpage is defined largely by our CSS
- Adherence to RESTful architectural principles
- Follows the MVC (model-view-controller) design pattern
  - Separation of the UI and data/logic by way of the controller
- Modular approach to allow for easier modifications and refactoring
  - Re-uses code and separates components
- Use of JavaScript to dynamically update page content without refreshing
- Uses a bearer token for authentication purposes
- Passwords are salted and hashed with SHA-1

3 Graphical User Interface



4 Design





5 Statistics

- 426 total commits and 6068 lines of code between 42 files make up CollegeChef
- 717 lines of HTML between 8 files
- 779 lines of CSS between 8 files
- 4572 lines of JavaScript between 26 files
  - 6 front-end files, 20 back-end files
  - 34 API endpoints
  - 11 data schemas
- Technologies used: W3.CSS framework, jQuery JS library, node.js for runtime, MongoDB for our database, and the following supporting NPM modules: body-parser, browser-sync, contributor, ejs, express, express-bearer-token, jwt-simple, mongod, mongodb, mongoose, mongoose-auto-increment, node.js, sha1

6 Future Work

In the future we would like to expand the scale of our web app. Our app functions well as is, but handling the sort of traffic required for a large user-base will hurt the user experience.

With a large user-base we would also need reliable hosting (e.g. a dedicated server) and a more thorough testing-suite such that the app will not easily crash. Better back-up solutions would also protect against this.

Although our passwords are salted and hashed with SHA-1, we can always make our application more secure as password cracking is not the only vulnerability of a web app. Better validation and protection against CSRF and XSS attacks is essential.

We would also like to expand the functionality itself to create a real community (e.g. with user messaging) where users can make new friends and interact with other food lovers.

7 References

1. CSC309 Winter 2017 lecture and lab material
2. https://www.w3schools.com/w3css/ (tutorials and documentation)
3. http://mongoosejs.com/docs/index.html (database documentation)
4. https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/ (database documentation)
5. https://www.npmjs.com/package/mongoose-auto-increment (id auto-increment documentation)
6. https://codepen.io/jamesbarnett/pen/vlpkh?editors=1000 (rating widget tutorial)
7. http://fontawesome.io/get-started/ (icon documentation)