**Database Application Programming III**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Milestone 2 | |
| **Date:** | 2/4/18 | |
| **Revision:** | 1.5 | |
| **Team:** | 1. Connor Low | |
| 2. Ali Cooper | |
| **Weekly Team Status Summary:** | |  |  |  | | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | | Updated sitemap, UML diagrams | CL/AC | 2 | | Updated Login and Registration. Implemented services. | CL | 5.5 | | Implemented Utility Classes | AC | 3 | | Created Administrator functionality (admin.php, AdminController.php, SuspendUser services) | CL | 6 | | Implemented Profile Business and Data Services | AC | 3 | | DB Design | AC | .5 | | Bootstrap integration | AC | 2 | | Delete user, $user object persistence in session, error messages | CL | 3.5 | | Added layouts and components | AC | 4 | | Added Particles JS | AC | .2 | | |
| **GIT URL:** | <https://github.com/n4n0byte/CST256.git>  Github invite: https://github.com/n4n0byte/CST256/invitations | |
| **Hosting URL:** |  | |
| **Cast url:** | **Part A:**https://www.useloom.com/share/77770093b1a247d28849a5d478e41583  **Part B:** https://www.useloom.com/share/46c917c10e5f4df8a7c1980bcc3679c0 | |
| **Peer Review:** | Y | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Design Documentation**

**Install Instructions:**

1. Pull project from Github
   1. If using git in bash, use *git clone https://github.com/n4n0byte/CST256.git* to clone the repository into a desired folder.
   2. If using a git GUI, paste *https://github.com/n4n0byte/CST256.git* into the clone input.
   3. *Project/CLC* is the main Laravel project directory.
2. Set up database
   1. In the *Documentation* repository directory, go into */DB\_Design* and locate the *larabar.sql*.
   2. Create a schema and import *larabar.sql* tables in MySQL Workbench:
      1. Establish a connection.
      2. Create a new schema (we suggest naming it “larabar”, but you may update the *.env* file in the main project directory to reflect any name you may chose).
      3. Navigate to *Data Import/Restore* under *Management*.
      4. For *Import Options*, select *Import from Self-Contained File* and enter the location of *larabar.sql*.
      5. Set *Default Target Schema* to the name of your database schema. Start Import.
      6. Confirm that *Import of [document\_location] has finished* is logged. Test the database with a SELECT statement.
   3. Navigate to the main project directory: *Project/CLC.*
      1. Open the *.env* file
      2. Set *DB\_USERNAME=root*
      3. Set *DB\_PASSWORD=root*

**General Technical Approach:**

Naming conventions:

* **Classes**: camelcase, starting with capitals (*ClassName*).
* **URL**: lowercase with “\_” as spaces between words (*another\_uri*)
* **Everything else**: camelcase (*varName*).

Stylesheets:

All stylesheets should be compiled from .*less* or pure .*css*. The main stylesheet is *styles.less* found in the *public/css* directory.

Database connection:

* Database, username, and password are set in .env
* Database name: *larabar*

Git:

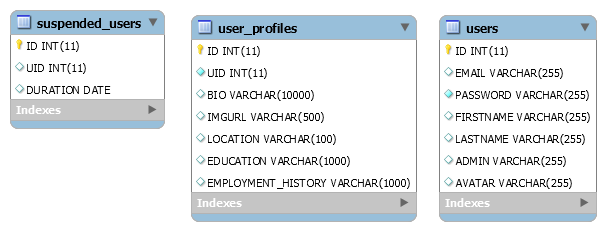
* Git ignore: environment and build files.

**Key Technical Design Decisions:**

Every table has matching model object, business service, controller, and data service to support the N-Layer standards. These are stored in the */src* folder.

User’s login-status is determined by the existence of a UserModel in the session.

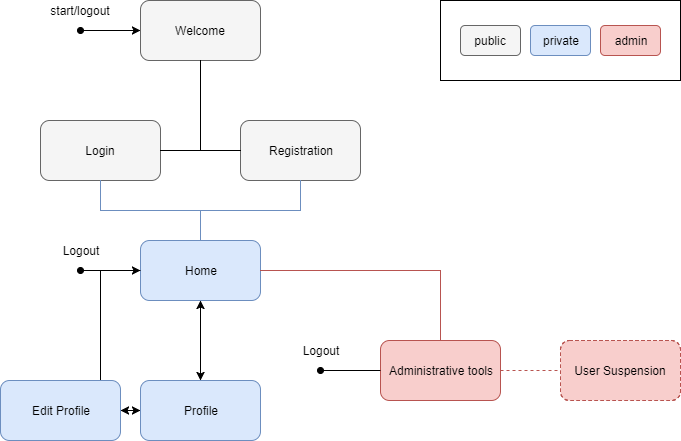
**ER Diagram:**



**DDL Scripts:**

https://github.com/n4n0byte/CST256/blob/master/Documentation/DB\_Design/larabar.sql

**Sitemap Diagram:**

****

**Security Design:**

The UserController.php currently handles form errors for Login and Registration. Actions will redirect to an error page if authentication fails.

User roles are determined by a boolean value in the database (*Admin*).

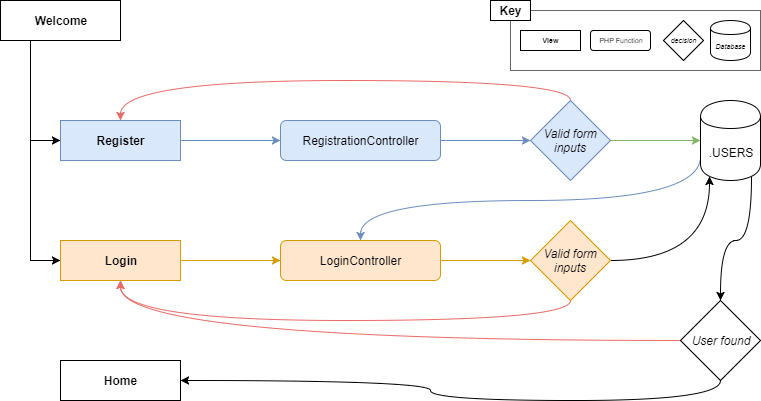
**3rd Party Interface Design:**

N/A

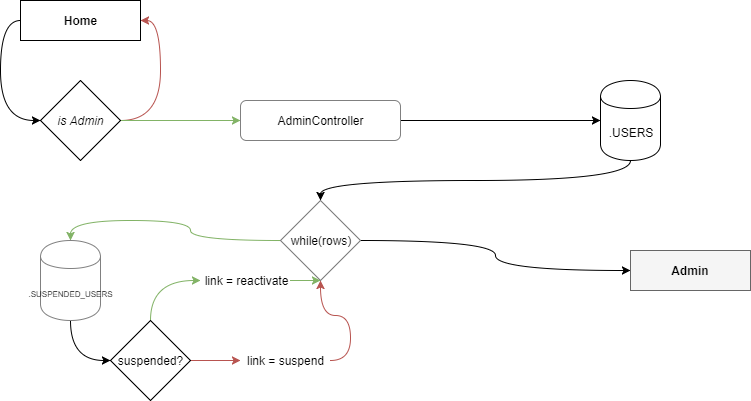
*This section should fully document any 3rd Party Service Interface API’s, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a 3rd party developer to integrate with the service and API.*

**Flow Charts:**

Authentication

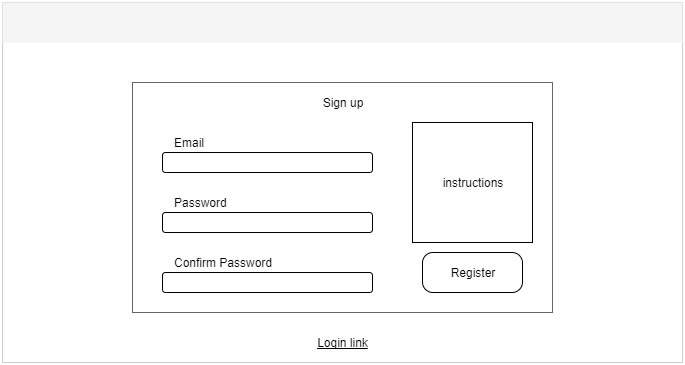


Admin: suspend/reactivate



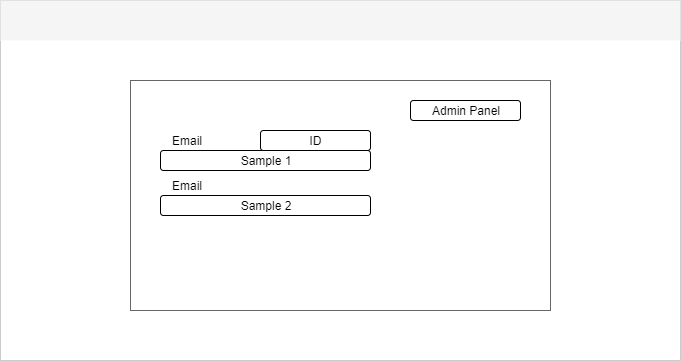
**User Interface Diagrams:**

Register

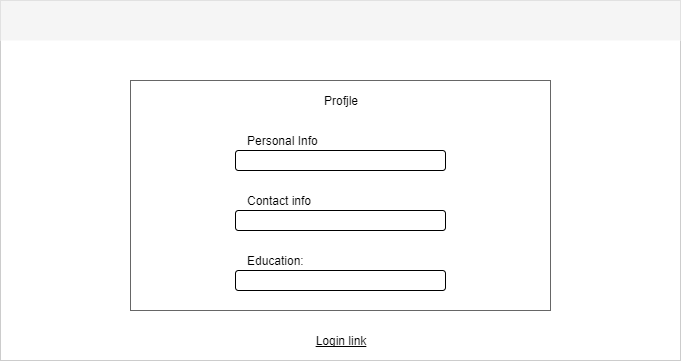


Login

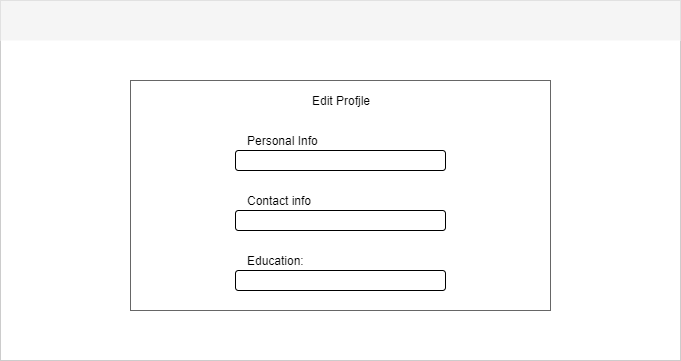




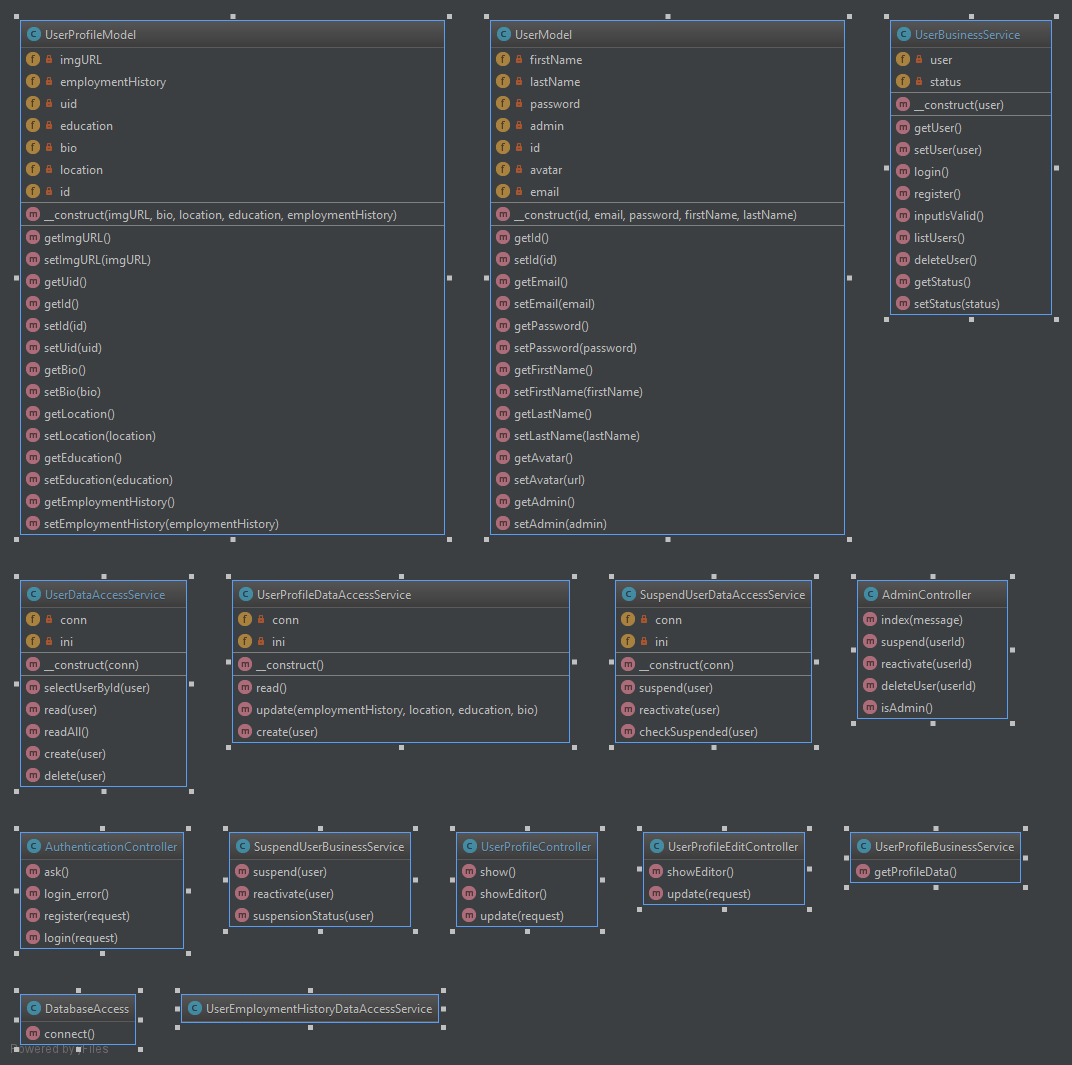
Profile



Edit Profile



**Class Diagrams:**



**Pseudo Code:**

Register - User clicks submit

* Get variables ($email, $password, $name)
* Create user object. Pass to Business Service.
  + Check input data validity. Pass to Data Service.
    - Check for existing Username. Update Database. Return result.
* If failure, direct to register form
* If success, login user. Redirect to home page.

Login - User clicks submit

* Get variables ($email, $password, $name)
* Create user object. Pass to Business Service.
  + Check input data validity. Pass to Data Service.
    - Check for existing Users. Update Database. Return result.
* If failure, direct to register form
* If success, login user. Redirect to home page.

**Other Documentation:**

**Risks:**

Unknown how to scaffold or auto-include models (and data services and business services) in controllers.

**Known Issues:**

* Most *vendor* sub-directories have *.gitignore* files that prevent the directory contents from pushing. The *vendor* directory must be copied from a pre-existing Laravel project.
* Bootstrap
* Particle.js not in git for some reason.