**Software Development Project**

**Deliverable 4**

**Group 20**

**Colm Le Gear – 15148823**

**David Kearney – 15169235**

**Liam Horgan – 14164736**

**Ryan Harris –**

Contents

[Introduction](#page3) [2](#page3)

[Major Processes & Feature List](#page4) [3](#page4)

[Registration & Create Profile](#page4) [3](#page4)

[Search](#page4) [3](#page4)

[Matching](#page5) [4](#page5)

[Messaging](#page5) [4](#page5)

[Admin](#page6) [5](#page6)

[Additional Features](#page6) [5](#page6)

[View Visitors](#page6) [5](#page6)

[Like User](#page6) [5](#page6)

[Hide User](#page6) [5](#page6)

[Block User](#page6) [5](#page6)

[See Profile as Other](#page6) [5](#page6)

[Edit Profile](#page6) [5](#page6)

[Reset Password](#page6) [5](#page6)

[Error Page](#page7) [6](#page7)

[Potential Additions](#page7) [6](#page7)

[Photo Upload](#page7) [6](#page7)

[Dynamic Question Generation](#page7) [6](#page7)

[Other Resources Used](#page7) [6](#page7)

[Bootstrap](#page7) [6](#page7)

[Bootstrap Select](#page7) [6](#page7)

[Font-Awesome](#page7) [6](#page7)

[Google Fonts](#page7) [6](#page7)

[Appendix](#page8) [7](#page8)

[User Site Files](#page8) [7](#page8)

[Admin Site Files](#page10) [9](#page10)

[Database E-R Diagram](#page11) [10](#page11)

[Database Create Statements](#page12) [11](#page12)

Introduction

This report presents the final state of the CS4014 Software Development project as completed by Group 20.

For the presentation, the group used a private web server, but the database was on the CSIS server provided. All files have since been copied to the group’s testweb folder, exactly as they were during the presentation.

This deliverable includes an outline of all major processes and features, identifying the files involved, and does the same for additional features present on the site. There is also a short section discussing worthwhile features that could potentially have been added if the time had been available, and also a discussion of how certain features could have been handled differently in retrospect.

The project can be cloned via github : https://github.com/CS4084/CS4084

Additional resources used in the creation of the site are listed, where we have relied on free third-party code for certain element of the site.

Appendixes are included listing all main files used in the user and admin sites, with a brief description of each file. They also include a database E-R diagram and create statements for all database tables, to indicate foreign keys and data types.

Major Processes & Feature List

Relevant pages or scripts are listed in square brackets.

All input submission includes both JavaScript and PHP validation, with all text input sanitised, and all dropdown input compared to acceptable range before being used in database queries.

All text input is handled to prevent script injection attacks using the htmlentities() PHP function, which also allows input with special characters, such as quotes, to be recorded in the database. When output, all such fields are stripped of escape characters.

Login and Registration

* Login form [**index.php** -[URL] []](http://176.32.230.49/meet-cute.com/index.php) requiring email address and password.
* If site visitor is not a member they can click a link that will bring them to the register page.
* Site visitor can create an account with [**register.php**].

1. User can enter their details such as name, student ID, email, password to be stored in the database.

* If user’s data is entered correctly (e.g. matching passwords and valid UL email address) then they will be redirected back to the login page and shown a registered successfully message.

Create Task

* Registered users can create a task using [**newtask.php**].
* This allows users to enter their task details and checks their input for errors. If their task is error free then the data are stored in the **task** table in the database.

Task List

* On the homepage [**dashboard.php**] and [**displaytask.php**] displays a task list showing all available tasks to the user.
* **All Tasks -** This will show details of all tasks that are available.
* **Subscribed Tags –** This will show tasks that include tags that users are subscribed to.
* **Most Viewed Tags –** This shows tasks that the user has viewed. The more they’ve viewed a task the higher up it will appear in this list.
* **My Subject Stream –** This shows tasks that belong to the user’s subject stream (faculty).

View Task Details

* [**task.php**] is used when a user clicks on a task to view more task details.
* This displays all task details.
* Allows user to claim task and download the document/document sample.
* Allows user to complete task using [**completetask.php**].
* Allows user to un-publish their own task and moderators to un-publish flagged tasks.

Reputation

* Reputation score is stored as **repScore** in the **users** table in the database**.** This score can be increased by claiming tasks, flagging tasks and receiving positive reviews on completed tasks. It can be decreased by failing to complete tasks, cancelling tasks and receiving a negative review for completing a task. If a user achieves a score of 40 or greater they are considered a moderator and have access to features such as viewing flagged tasks, un-publishing tasks and banning users.

Profile

* [**profile.php**] displays user’s details
* A moderator has the ability to ban user using [**ban.php**] from the profile page. Note: The ban user button is not visible to users who are not moderators.
* Users can click a link on this page that redirects them to the user’s available tasks. If the user is on their own profile page then they will be redirected to **Open Tasks** in the **My Tasks** section. This uses [**usertasks.php**].

Ban User

* A moderator can ban a member by viewing their profile and clicking the “Ban User” button which uses [**ban.php**] . Their email will be blacklisted from the site so that they won’t be able to login or register using that email again and the bannedUser ID and moderator ID will be stored in the **user\_banned** table.

Flagged Tasks Page

* Only moderators can access this section. They will be able to see a list of all flagged tasks. From this page they can view the profile of the user who created the flagged task and ban them.

Un-Publish Task

* Moderators can un-publish flagged tasks using [**task.php**]. The un-published task is removed from open tasks and added to the **unpublished\_tasks** table in the database.

Search

* Searches are performed using [**search.php**]
* This php file runs an SQL query that returns available tasks (not claimed, not un-published, not completed) that match the search string.
* [**searchpredict.php**] is used as the source for the JQuery autocomplete function in the search box. This queries the database to find the matching results and display them underneath the search text box.

Potential Additions

Photo Upload – our group had prepared the script to handle this [**upload.php**] and relevant sections were originally included on the create-profile.php form, including JavaScript validation of file type. However, in the end there was not sufficient time to test the feature before the presentation, and so it was decided to remove it rather than risk potential problems.

Dynamic Question Generation –the database has been designed to allow for dynamic question display, where the html to display the question could have been generated automatically by calling on the database. The question table includes an attribute to specify the html form input type, which could have been used to determine the style of question to present. However, while all question options are populated dynamically from the database, we did not have time to realise this process fully for questions, and so the question itself was hard-coded in the html.

Other Resources Used

The following are various resources used in the creation of our site. Files for these resources

Bootstrap – the front-end of the site was built using Bootstrap 3, with a large degree of customisation of styling and features. For example, the basic Bootstrap tab component was reworked to form the basis of the inbox in the message.php and reports.php pages, or the collapse component was reworked to add button navigation for the form on create-profile.php. The files required to support Bootstrap are included in the **css** and **js** subfolders of the main site.

Bootstrap Select – this plugin was used to create multi-select dropdown menus. It adds files to the **css** and **js** subfolders of the main site.

Font-Awesome – this plugin was used to add certain icons. It is included using a CDN link at the top of the relevant display pages.

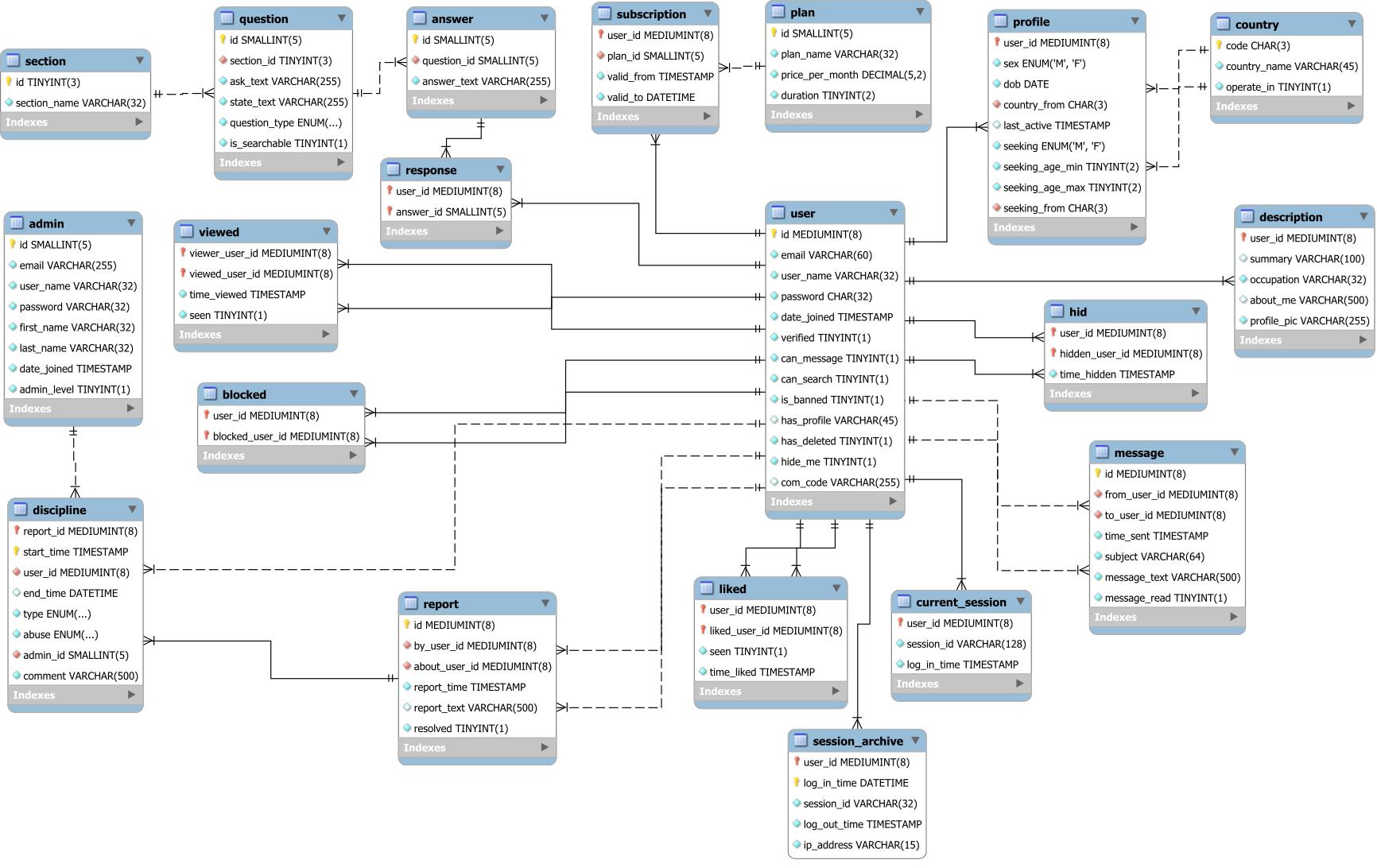
Appendix

Site Files

The following are all display pages and scripts used in the user section of the site. CSS and JS file are not listed in detail, though relevant CSS files are mentioned in the display page listing. No external JavaScript files were created by the group for this project, so all JavaScript files are part of 3rd-party components, such as Bootstrap.

|  |  |
| --- | --- |
| **FILE** | **DESCRIPTION** |
| ban.php | Script for banning users via their profile page [**profile.php**]. |
|  |  |
|  |  |
| claimedtasks.php | Script for user to view their claimed tasks. |
|  |  |
|  |  |
| completetask.php | Script for completion of a task. Inserts task completion data such as the review, into **task\_completed** table in the database. |
|  |  |
| config.php | Script to connect to database and sets it to variable **$db.** |
|  |  |
| dashboard.php | Contains SQL statements to display all available tasks, subscribed tags tasks, most viewed tasks and faculty/subject tasks. |
|  |  |
| displaytasks.php | Creates HTML elements for displaying tasks and optionally removes tasks if they are past the claim deadline. |
|  |  |
| flag.php | Script to check input from subscribe.php and credit card validation. |
|  |  |
| flaggedtasks.php | Script to use as include on all pages, containing function for database connection, |
|  | request and connection close. |
|  |  |
| index.php | Script containing database login variables. |
|  |  |
| logout.php | Special script created for testing, to allow testers to quickly delete an account from |
|  | all tables, allowing a previously registered email address to be reused. Not for |
|  | inclusion in site features. |
|  |  |
| newtasks.php | Page to display error messages resulting from PHP validation. Styled in error.css |
|  |  |
| profile.php | send to search-results.php for display. |
|  |  |
| register.php | Script to handle login from the verify-email.php page. Includes check that unique |
|  | url of the page (using a GET variable) matches that recorded in the database for |
|  | that user. |
|  |  |
| search.php | Script to let user see all the members they have hidden. Creates array of results to |
|  | send to search-results.php for display. |
|  |  |
| searchpredict.php | Script that hides a specified user. |
|  |  |
| tag.php | Page that acts as hub. Users directed here after successful login, and suggested |
|  | matches displayed here. Includes search features. Styled in home.css |
|  |  |
| tagsearch.php | Landing page for login / registration. Checks if user is already logged in, and if so |
|  | redirects them to home.php, to prevent double login problems. Styled in index.css |
|  |  |
| task.php | Script to let user see all the members they have liked. Creates array of results to |
|  | send to search-results.php for display. |
|  |  |
| unflag.php | Script that likes a specified user. |
|  |  |
| usertasks.php | Script that handles login through the normal login form on index.php. Includes |
|  | option for long-term cookie creation. |
|  |  |
| verify.php | Script to logout user, destroying all cookies. |
|  |  |

Database E-R Diagram



|  |
| --- |
| 10 |

Database Create Statements

The following is a list of create statements for all database tables in alphabetic order. Not all tables or attributes presented here were used in the final implementation of the site.

CREATE TABLE **admin** (

id smallint(5) unsigned NOT NULL AUTO\_INCREMENT, email varchar(255) NOT NULL,

user\_name varchar(32) NOT NULL, password varchar(32) NOT NULL, first\_name varchar(32) NOT NULL, last\_name varchar(32) NOT NULL,

date\_joined timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP, admin\_level tinyint(1) unsigned NOT NULL DEFAULT '0', PRIMARY KEY (id),

UNIQUE KEY user\_name\_UNIQUE (user\_name), UNIQUE KEY email\_UNIQUE (email)

);

CREATE TABLE **answer** (

id smallint(5) unsigned NOT NULL AUTO\_INCREMENT, question\_id smallint(5) unsigned NOT NULL, answer\_text varchar(255) NOT NULL,

PRIMARY KEY (id),

KEY answer\_question\_idx (question\_id),

CONSTRAINT answer\_question FOREIGN KEY (question\_id) REFERENCES question (id) ON DELETE

CASCADE ON UPDATE CASCADE );

CREATE TABLE **blocked** (

user\_id mediumint(8) unsigned NOT NULL, blocked\_user\_id mediumint(8) unsigned NOT NULL, PRIMARY KEY (user\_id,blocked\_user\_id),

KEY block\_uid\_2\_idx (blocked\_user\_id),

CONSTRAINT block\_uid\_1 FOREIGN KEY (user\_id) REFERENCES user (id) ON DELETE CASCADE, CONSTRAINT block\_uid\_2 FOREIGN KEY (blocked\_user\_id) REFERENCES user (id) ON DELETE CASCADE

);

CREATE TABLE **country** ( code char(3) NOT NULL,

country\_name varchar(45) NOT NULL,

operate\_in tinyint(1) unsigned NOT NULL DEFAULT '0', PRIMARY KEY (code),

UNIQUE KEY country\_name\_UNIQUE (country\_name)

);

CREATE TABLE **credit\_card** (

user\_id mediumint(8) unsigned NOT NULL, name\_on\_card varchar(64) NOT NULL, card\_number varchar(16) NOT NULL, expiry\_month tinyint(2) unsigned NOT NULL, expiry\_year tinyint(2) unsigned NOT NULL, sec\_code tinyint(3) unsigned NOT NULL, PRIMARY KEY (user\_id),

CONSTRAINT credit\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **current\_session** (

user\_id mediumint(8) unsigned NOT NULL, session\_id varchar(128) NOT NULL,

log\_in\_time timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP, PRIMARY KEY (user\_id),

KEY session\_id (session\_id),

CONSTRAINT active\_sess\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **description** (

user\_id mediumint(8) unsigned NOT NULL,

summary varchar(100) DEFAULT 'I\\''m new to the site, so why not message me to say hello.', occupation varchar(32) NOT NULL,

about\_me varchar(500) DEFAULT 'I''m new and still working on how to introduce myself, but in the meantime message me if you''d like to find out more.',

profile\_pic varchar(255) NOT NULL DEFAULT 'images/profile\_pics/pic\_gm3.png', PRIMARY KEY (user\_id),

CONSTRAINT desc\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **discipline** (

report\_id mediumint(8) unsigned NOT NULL,

start\_time timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, user\_id mediumint(8) unsigned NOT NULL,

end\_time datetime DEFAULT NULL,

type enum('warning','message\_ban','full\_ban','none') NOT NULL DEFAULT 'none', abuse

enum('advertising','false\_identification','harassment','offensive\_content','spam','none') NOT NULL DEFAULT 'none',

admin\_id smallint(5) unsigned NOT NULL,

comment varchar(500) NOT NULL DEFAULT 'No Action Taken', PRIMARY KEY (report\_id,start\_time),

KEY disc\_uid\_idx (user\_id), KEY disc\_admin\_idx (admin\_id),

CONSTRAINT disc\_admin FOREIGN KEY (admin\_id) REFERENCES admin (id) ON UPDATE CASCADE, CONSTRAINT disc\_report FOREIGN KEY (report\_id) REFERENCES report (id) ON UPDATE CASCADE, CONSTRAINT disc\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **extra\_pic** (

user\_id mediumint(8) unsigned NOT NULL, pic\_1 varchar(255) DEFAULT NULL,

pic\_2 varchar(255) DEFAULT NULL, pic\_3 varchar(255) DEFAULT NULL, pic\_4 varchar(255) DEFAULT NULL, pic\_5 varchar(255) DEFAULT NULL, pic\_6 varchar(255) DEFAULT NULL, PRIMARY KEY (user\_id),

CONSTRAINT extra\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **hid** (

user\_id mediumint(8) unsigned NOT NULL, hidden\_user\_id mediumint(8) unsigned NOT NULL,

time\_hidden timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, PRIMARY KEY (hidden\_user\_id,user\_id),

KEY hid\_uid\_1\_idx (user\_id),

CONSTRAINT hid\_uid\_1 FOREIGN KEY (user\_id) REFERENCES user (id) ON DELETE CASCADE, CONSTRAINT hid\_uid\_2 FOREIGN KEY (hidden\_user\_id) REFERENCES user (id) ON DELETE CASCADE

);

CREATE TABLE **liked** (

user\_id mediumint(8) unsigned NOT NULL, liked\_user\_id mediumint(8) unsigned NOT NULL, seen tinyint(1) unsigned NOT NULL DEFAULT '0',

time\_liked timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, PRIMARY KEY (user\_id,liked\_user\_id),

KEY liked\_uid\_2\_idx (liked\_user\_id),

CONSTRAINT liked\_uid\_1 FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE, CONSTRAINT liked\_uid\_2 FOREIGN KEY (liked\_user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **message** (

id mediumint(8) unsigned NOT NULL AUTO\_INCREMENT, from\_user\_id mediumint(8) unsigned NOT NULL, to\_user\_id mediumint(8) unsigned NOT NULL,

time\_sent timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP, subject varchar(64) NOT NULL DEFAULT 'RE:', message\_text varchar(500) NOT NULL,

message\_read tinyint(1) unsigned NOT NULL DEFAULT '0', PRIMARY KEY (id),

KEY message\_uid\_1\_idx (from\_user\_id), KEY message\_uid\_2\_idx (to\_user\_id),

CONSTRAINT message\_uid\_1 FOREIGN KEY (from\_user\_id) REFERENCES user (id) ON UPDATE CASCADE, CONSTRAINT message\_uid\_2 FOREIGN KEY (to\_user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **plan** (

id smallint(5) unsigned NOT NULL AUTO\_INCREMENT, plan\_name varchar(32) NOT NULL,

price\_per\_month decimal(5,2) unsigned NOT NULL,

duration tinyint(2) NOT NULL COMMENT 'Length of plan in months. Default is 1 month. -1 for infinite time',

PRIMARY KEY (id)

);

CREATE TABLE **profile** (

user\_id mediumint(8) unsigned NOT NULL, sex enum('M','F') NOT NULL DEFAULT 'M', dob date NOT NULL,

country\_from char(3) NOT NULL, last\_active timestamp NULL DEFAULT NULL, seeking enum('M','F') NOT NULL DEFAULT 'F',

seeking\_age\_min tinyint(2) unsigned NOT NULL DEFAULT '18', seeking\_age\_max tinyint(2) unsigned NOT NULL DEFAULT '99', seeking\_from char(3) NOT NULL,

PRIMARY KEY (user\_id),

KEY prof\_country\_from (country\_from), KEY prof\_country\_seek (seeking\_from),

CONSTRAINT prof\_country\_from FOREIGN KEY (country\_from) REFERENCES country (code) ON UPDATE

CASCADE,

CONSTRAINT prof\_seeking\_from FOREIGN KEY (seeking\_from) REFERENCES country (code) ON UPDATE

CASCADE,

CONSTRAINT prof\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON DELETE NO ACTION ON UPDATE

NO ACTION );

CREATE TABLE **question** (

id smallint(5) unsigned NOT NULL AUTO\_INCREMENT, section\_id tinyint(3) unsigned NOT NULL,

ask\_text varchar(255) NOT NULL, state\_text varchar(255) NOT NULL, question\_type

enum('select','multi\_select','checkbox','radio\_5','radio\_3','text\_input','paragraph','toggle')

NOT NULL,

is\_searchable tinyint(1) unsigned NOT NULL DEFAULT '1', PRIMARY KEY (id),

KEY quest\_section\_idx (section\_id),

CONSTRAINT quest\_section FOREIGN KEY (section\_id) REFERENCES section (id) ON UPDATE CASCADE

);

CREATE TABLE **report** (

id mediumint(8) unsigned NOT NULL AUTO\_INCREMENT, by\_user\_id mediumint(8) unsigned NOT NULL, about\_user\_id mediumint(8) unsigned NOT NULL, report\_time timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP, report\_text varchar(500) DEFAULT NULL,

resolved tinyint(1) unsigned NOT NULL DEFAULT '0', PRIMARY KEY (id),

KEY report\_uid\_1\_idx (by\_user\_id), KEY report\_uid\_2\_idx (about\_user\_id),

CONSTRAINT report\_uid\_1 FOREIGN KEY (by\_user\_id) REFERENCES user (id) ON UPDATE CASCADE, CONSTRAINT report\_uid\_2 FOREIGN KEY (about\_user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **response** (

user\_id mediumint(8) unsigned NOT NULL, answer\_id smallint(5) unsigned NOT NULL, PRIMARY KEY (user\_id,answer\_id),

KEY response\_answer\_id\_idx (answer\_id),

CONSTRAINT response\_answer\_id FOREIGN KEY (answer\_id) REFERENCES answer (id) ON DELETE

CASCADE ON UPDATE CASCADE,

CONSTRAINT response\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **section** (

id tinyint(3) unsigned NOT NULL, section\_name varchar(32) NOT NULL, PRIMARY KEY (id)

);

CREATE TABLE **session\_archive** (

user\_id mediumint(8) unsigned NOT NULL, log\_in\_time datetime NOT NULL, session\_id varchar(32) NOT NULL,

log\_out\_time timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP, ip\_address varchar(15) NOT NULL,

PRIMARY KEY (user\_id,log\_in\_time),

CONSTRAINT archive\_user\_id FOREIGN KEY (user\_id) REFERENCES user (id) ON DELETE NO ACTION ON

UPDATE NO ACTION );

CREATE TABLE **subscription** (

user\_id mediumint(8) unsigned NOT NULL, plan\_id smallint(5) unsigned NOT NULL,

valid\_from timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, valid\_to datetime NOT NULL,

PRIMARY KEY (user\_id),

KEY sub\_plan\_idx (plan\_id),

CONSTRAINT sub\_plan FOREIGN KEY (plan\_id) REFERENCES plan (id) ON UPDATE CASCADE, CONSTRAINT sub\_uid FOREIGN KEY (user\_id) REFERENCES user (id) ON UPDATE CASCADE

);

CREATE TABLE **user** (

id mediumint(8) unsigned NOT NULL AUTO\_INCREMENT, email varchar(60) NOT NULL,

user\_name varchar(32) NOT NULL, password char(32) NOT NULL,

date\_joined timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP, verified tinyint(1) unsigned NOT NULL DEFAULT '0', can\_message tinyint(1) unsigned NOT NULL DEFAULT '0', can\_search tinyint(1) unsigned NOT NULL DEFAULT '0', is\_banned tinyint(1) unsigned NOT NULL DEFAULT '0', has\_profile varchar(45) DEFAULT '0',

has\_deleted tinyint(1) unsigned NOT NULL DEFAULT '0', hide\_me tinyint(1) unsigned NOT NULL DEFAULT '0', com\_code varchar(255) DEFAULT NULL,

PRIMARY KEY (id),

UNIQUE KEY email\_UNIQUE (email),

UNIQUE KEY user\_name\_UNIQUE (user\_name)

);

CREATE TABLE **viewed** (

viewer\_user\_id mediumint(8) unsigned NOT NULL, viewed\_user\_id mediumint(8) unsigned NOT NULL,

time\_viewed timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, seen tinyint(1) unsigned NOT NULL DEFAULT '0',

PRIMARY KEY (viewer\_user\_id,viewed\_user\_id), KEY viewed\_uid\_2\_idx (viewed\_user\_id),

CONSTRAINT FK\_viewed\_user FOREIGN KEY (viewer\_user\_id) REFERENCES user (id) ON UPDATE

CASCADE,

CONSTRAINT viewed\_uid\_2 FOREIGN KEY (viewed\_user\_id) REFERENCES user (id) ON UPDATE CASCADE

);