System Overview for Advanced Development Assignment

A. Access and Testing

Test Account Details

Student email: testi7648171@gmail.com, Name: Test Student, Password: Testpass123
Tutor email: tutori7648171@gmail.com, Name: Test Tutor, Password: Testpass123
No account email: nodatai7648171@gmail.com, Password Testpass123
Site: https://i7648171.appspot.com
Trello board: https://trello.com/invite/b/VtRq7f1I/beb990cc1cc7b9054b1e8649b146a9e6/mitigating-circumstances
App engine (tutor email has read access): https://console.cloud.google.com/home/dashboard?project=i7648171

Submission test

1. Go to https://i7648171.appspot.com and login using the student email and password.
2. Click the “SUBMIT REQUEST” button in the top left.
3. Fill out the details then click submit, image is optional, other fields are not.
4. If successful you should be redirected to the new request, tutors will be emailed about the new request and a trello card will be created for it.

View request as student test

1. Login as a student.
2. Click the MY OPEN REQUESTS button if not already on the requests page.
3. If there is no open request for the student then make a new one, else click the title of the request.
4. If successful you should be on the page for the request with its details and the options to submit a follow-up or close the request.

Change assigned tutor test

1. As a tutor go to a request that is not assigned to the tutor.
2. Under “Assigned To” there should be a drop down, change it to the tutor.
3. If successful the request should refresh with the new tutor being the drop-down value and an email sent to the old and new tutors to inform them of the change, if the request was unassigned to changed to unassigned then the trello card for it should change list.

Submit follow-up on request test

1. As a student or tutor go to a request, if a tutor then make sure the request is assigned to the tutor.
2. Try to submit a follow-up for the request, the form is under the “Submit Followup” text, text is mandatory and image is optional.
3. If text is accepted and optional image is valid then the follow-up should submit and the page refresh to show the new follow-up, if the request has a student and tutor then the one that did not make the follow-up should receive an email notifying them that a follow up has been submitted and the trello card should have the comment added to it.

Close request test

1. As a student or tutor go to a request, if a tutor then make sure the request is assigned to the tutor.
2. Fill out the Close Request form then submit, text is mandatory.
3. If successful then the follow-up and close forms should disappear and the requests status should change from Open to a closed status, if tutor then it will be the reason you picked and if student then it will be “Closed by student”, tutor and or student will be emailed about the close and the trello card moved to the Closed list.

Request filter test

1. When logged in as a tutor go to https://i7648171.appspot.com/Requests or click either of the UNASSIGNED REQUESTS or MY REQUESTS buttons.
2. Change the status and or tutor drop downs to a desired combination then click submit.

3. If there is data for the combination then you should see only the selected combination of request types.

Add new user test

1. Login as a tutor then click the “ADD A NEW USER” button.
2. Enter a name, email address and choose a role then submit.
3. If successful then you should be redirected to the new user’s page and they should receive an email informing about the account being made for them.

Access site with no account test

1. Logout of the current account if any.
2. Login using the “No account” email.
3. If successful you should be redirected to a page with a logout button and text saying to contact a tutor.

B. Files and Resources

1. Static folder:

SkeletonCSS folder contain the CSS files used for the look of the website.

Custom.css file for limiting the size of an image element using a class.

imagePreview.js which takes an image from an input element and display a preview of it in an img element.

textAreaGrow.js for increase the height of a text area as contents are added.

2. Vendor folder

Contains all the dependencies needed for the project and composer files for updating and managing dependencies.

3. Root directory files

.gcloudignore

Used during upload to the app engine to determine which files to not upload.

App.yaml

Contains the application name, version, php version, api version, mail handler and directory handlers for referring to files without using the relative address, i.e. /index instead of ../../View/index.html.

Composer.json

References for the required packages for the project, used by composer to determine what it should install and update.

Composer.lock

Locks all the dependencies of the project to a known state.

Index.yaml

Contains the indexes used for Google datastore queries that get multiple values and or have multiple conditions as they require an index.

4. StudMitSys/Control folder

Contains most of the logic code.

emailFunctions.php

contains a function for setting up and sending an email using the app engine mail service. Email recipients, subject and email body are passed in when the function is called.

incomingEmailRequest.php

Handles emails sent to any address ending in @i7648171.appspotmail.com, if an email is sent to newRequest@i7648171.appspotmail.com from a student email then it makes a new request using the email details and if not a student then gives a response saying that only students can make requests.

requestDetails.php

Handles the non-static elements and procedures for the requestDetails.html page.
Requires a valid request ID be passed by get request as it will check if the user is a tutor or the student who made the request, if no then it will redirect back to the home page and if yes it will populate the page with data from the datastore and change page elements depending on if the user is a tutor or student.

requestDetailsFunctions.php

Handles AJAX post requests sent from the requestDetails page and returns a json encoded response, requires an ID for the request in question and can do 3 things:

1. Add a follow-up comment to a request if the data has been entered on the page and the user is the student that made or tutor assigned to the request, otherwise it will return a messaging saying which field is empty or another error condition, the follow can include an image which is stored in the google Storage bucket with a link in the datastore. If the follow up is successfully added, then the student or tutor will be notified of the follow-up by email and the request card on trello being updated with the follow-up text.
2. Change the tutor assigned to the request, only a tutor can do this and once done the tutors involved are emailed the change along with the trello request card being changed to Unassigned or Assigned if applicable.
3. Close a request, only by the assigned tutor or student with details changing for either, emailing the one that did not close it and updating the trello card to be closed.

Requests.php

Gets the data from datastore and builds page elements depending on if user is a student or a tutor and can filter the requests depending on the results of a get request, i.e. only open requests or only requests assigned to the logged in tutor etc.

smartyFunctions.php

Setups the smarty template engine for use on google drive, uses a storage bucket to store smarty data in a Storage bucket, this cannot be done locally on standard environment. Also contains functions for displaying the smarty header and making an email body using a template (tpl) file and passed in variables.

submitRequest.php

Setting up the submitRequest page, is only used to get the smarty header.

submitRequestFunctions.php

Handles the submitting of a new request through an AJAX post request, will add data to datastore if mandatory fields are not empty and if user, else will return an error message depending on the problem, optional submission of an image which goes to a storage bucket with a link to the image in the datastore.
Tutors will be notified by email of a new request and a trello card will be made with the request details.

trelloFunctions.php

Setting up a connection to the trello API and functions to: create cards, add comments to cards, getting a cards list ID and change a cards list/type using a card ID and other variables, when a new request is made a trello card is created for it.
Uses the Stevenmaguire Trello PHP wrapper and GuzzleHTTP for connecting and making changes through the trello API.

userDetails.php

Gets the data for the userDetails page depending on if the user is a student or a tutor, tutors can see anyone’s page, but a student can only see their own.

userFunctions.php

Functions used by other files and procedures to do various things with the google UserService API like:
Getting the group of the current user or email address,
Getting a user’s email, nickname, fullname.
Generating a logout URL.
Checking if the current user owns or is assigned to a request.
Also redirects to noAccount.html if the user does not have a role.

newUser.php

Sets up the newUser page, is only used to get the smarty header and redirect if user is not a tutor

newUserFunctions.php

Handles the submitting of a new user through an AJAX post request, will add data to datastore if mandatory fields are not empty and if user is a tutor, else will return an error message depending on the problem.

noAccount.php

Handles a use with no role on the system and displays a logout button.

5. StudMitSys/Model folder

Contains php code for interacting with the google datastore and cloudstore databases.

cloudDatastore.php

Sets up the connection to the google cloud datastore bucket and contains callable functions for reading and writing images to and from the bucket

readDatastore.php

Functions for getting data from the datastore using the cloud datastore and Tom Walder php-gds packages, two are used as the default interface for PHP to the datastore is a bit lacking/underdeveloped so php-gds is used in some cases.
Has functions for: getting data using a GQL query, single value using kind name and key and all values connected to a key in its kind.

writeDatastore.php

Setups connection to datastore and contains functions for: adding a new entity as a child of an existing entity, adding a new entity to a kind, updating a single existing value, upserting (updating and inserting where does not exist) multiple values and updating multiple existing values.

6. StudMitSys/Plugins folder

Contains plugins for the smarty template engine, plugins allow for php code to be used within template files. function.navbar.php is the only function used and creates the buttons for the page header depending on if the user is a student or tutor.

7. StudMitSys/templates folder

Contains smarty template files for smarty to build pages and emails.

emailNewRequest.tpl
Email that is sent to tutors when a new request is made

emailNotAStudent.tpl
Sent when a request email is received that does not come from a student.

emailNoTextOrTitle.tpl.
Sent when a request email is received for which the subject or email body is empty.

emailRequestClose.tpl
Sent to the student that made the request or the tutor assigned to the request when a request is closed by the other one, if tutor closes then student is emailed and vice versa.

emailRequestFollowup.tpl
Sent to the student that made the request or the tutor assigned to the request when a follow-up is submitted by the other one.

emailRequestSubmitted.tpl
Sent to the student when they successfully make a request.

emailTutorChange.tpl
Sent to the previously assigned and newly assigned tutors when the tutor for a request is changed.

footer.tpl
Footer file that contains closing div container, body and HTML tags.

header.tpl
Header file that contains the opening div container, body and HTML tags, SkeletonCSS references along with the head and header HTML elements.

emailNewUser.tpl
Sent to a new user when an account is made for them.

8. StudMitSys/View folder

Contains the frontend HTML elements and AJAX code.

requestDetails.html
Page for viewing the details of a request, submitting follow up comments, changing the assigned tutor and closing the request.
requestDetailsCloseForm.html
Page and form for a closing a request by AJAX post, is included in the page by requestDetails.php if the user is the assigned tutor or student that made the request.
requestDetailsFollowupForm.html
Page and form for submitting a request by AJAX post, is included in the page by requestDetails.php if the user is the assigned tutor or student that made the request.

Requests.html
Home page that shows all the requests depending on a get request with links to each request on the page, i.e. if get request with type=Open and tutor=NA then will get all open requests.
submitRequests.html
Page and form for a student to submit a new request by AJAX post, only accessible if user is a student.
userDetails.html
Shows details of a user and the requests owned by / assigned to them, a student can only view their own page and a tutor can see anyone's page.
newUser.html
Page and form for a tutor to submit a new user account, only accessible to tutor.
noAccount.html
Page for if the user does not have a role/account, informs them to contact a tutor and displays a logout button.

C. Main Dependencies

google/cloud-datastore v1.5.10 (Google 2018g)
Official Google PHP client for interfacing with the google NoSQL datastore, used as the database for non-file data.

tomwalder/php-gds v4.1.1 (Walder 2018)
Secondary client for interfacing with the datastore, used as I found some of the functionality of cloud-datastore to be lacking, for example when getting a date from the datastore using the official client it will get a DateTimeImmutable while php-gds returns a DateTime object which is easier to use.

smarty/smarty v3.1.33 (Ohrt 2018)
A PHP template engine that I used for template emails, header and footer.
Used as when looking for a template engine to use I found the most examples of it running on the app engine in comparison to Twig, Blade and Dawoo and was able to get it working on the app engine.

google/cloud-storage v1.9.1 (Google 2018f)
Official Google PHP client for interfacing with the google CloudStore, used for image storage.

floriansemm/official-library-php-email-parser 1.0.0 (Occhipinti 2014)
Library for parsing PHP MIME email data.
Used as the google recommended method of Mailparse (Google 2018e) is lacking in documentation making using it difficult.

stevenmaguire/trello-php 0.5.3 (Maguire 2018)
PHP wrapper for the trello API.
The default method of interacting with the API is using cURL, for unknown reasons when I enabled cURL on my app it slowed down to the point where a page takes around 4 minutes to load, there were no errors in any logs to indicate the cause and no code was changed.
This wrapper uses GuzzleHTTP instead of cURL so the slowdown is avoided.

D. Assignment Requirements

Template System
Smarty used for email templates, header and footer.

Students able to submit requests, evidence and keep track of requests
Students can submit request text, images and follow up comments using the form on the website, and submit requests by email to newRequest@i7648171.appspotmail.com.
Requests can be viewed by students on the site and any updates/comments made by tutors are viewable.

Tutors alerted when a new application is added
Tutors marked as active in the datastore are emailed when a new request is made.

Tutors able to review requests and evidence, update their status and contact students.
Tutors can see the request text and images that students submit, change the status of a request (closed, open, accepted etc) and any submit follow up comments which the student is notified about by email.

Use of OAuth/OpenID Connect
The login system used is the (Google 2018d) which uses OAuth 2.0 and Open ID Connect protocols, this is implemented in app.yaml through the “login: required” and “secure: always” lines and the app engines UserService.

AJAX Usage
AJAX is used for sending form data POST requests without refreshing on each page that uses a form to submit data. See requestDetails.html and submitRequest.html.

External API Interaction
When a request is made a trello card with its details is created and assigned to the “Unassigned” list, when a comment is made on request the trello card is updated with the comment and when a request is assigned to a tutor or closed the card is moved to the appropriate list. See trelloFunctions.php and its functions in submitRequestFunctions.php and requestDetailsFunctions.php.
Mitigating Circumstances Trello board:
https://trello.com/invite/b/VtRq7f1I/beb990cc1cc7b9054b1e8649b146a9e6/mitigating-circumstances

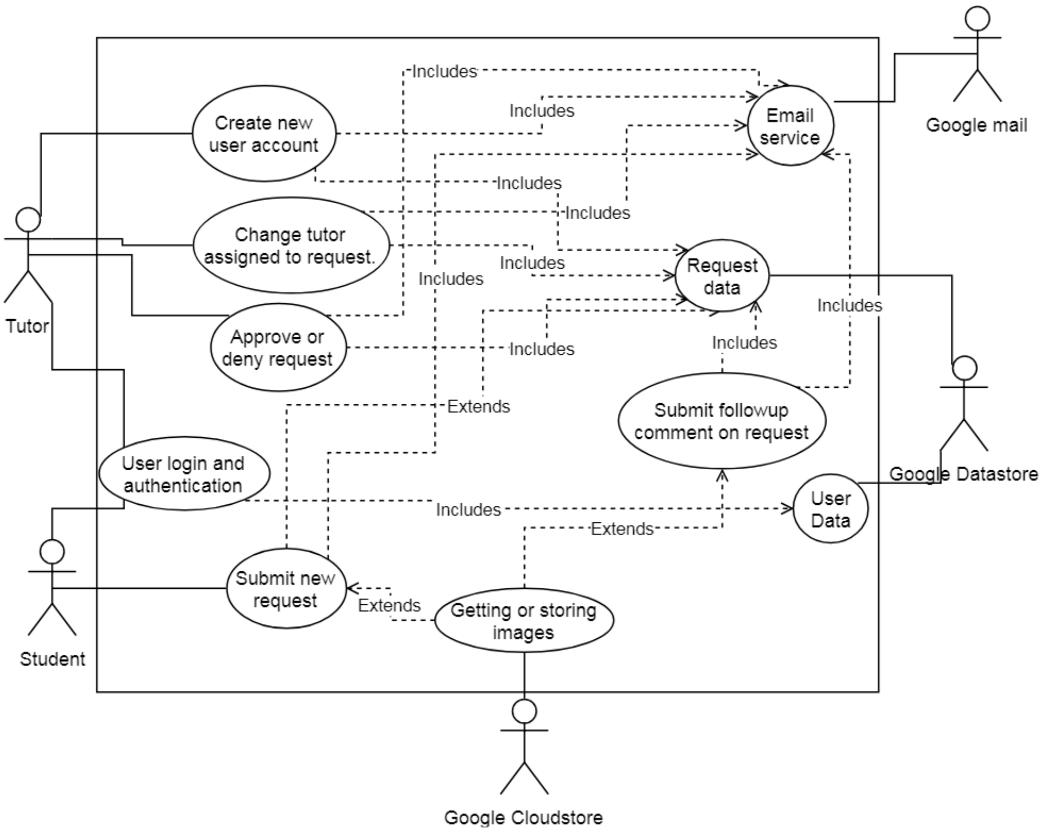
Native provider/cloud services used
App engine UserService (Google 2017)
NoSQL cloud Datastore (Google 2018c)
File CloudStore (Google 2018b)
App engine mail service (Google 2018a)
Trello API

Security Features
Login system using google user service (Google 2017).
Group system so that students cannot see what tutors and other students see, present in multiple pages and userFunctions.php
Login set to Required and secure set to Always on all user exposed pages, see app.yaml (Google 2018h).

Mobile compatibility
Site and operations were tested on an android device using Chromium and Firefox 64.0.1, nothing was found that prevented the site from being usable.

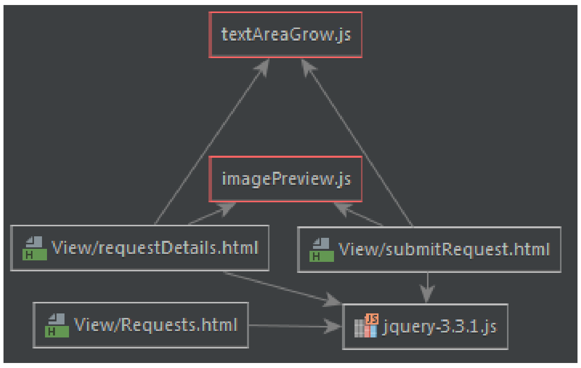
E. Diagrams

System Use Case



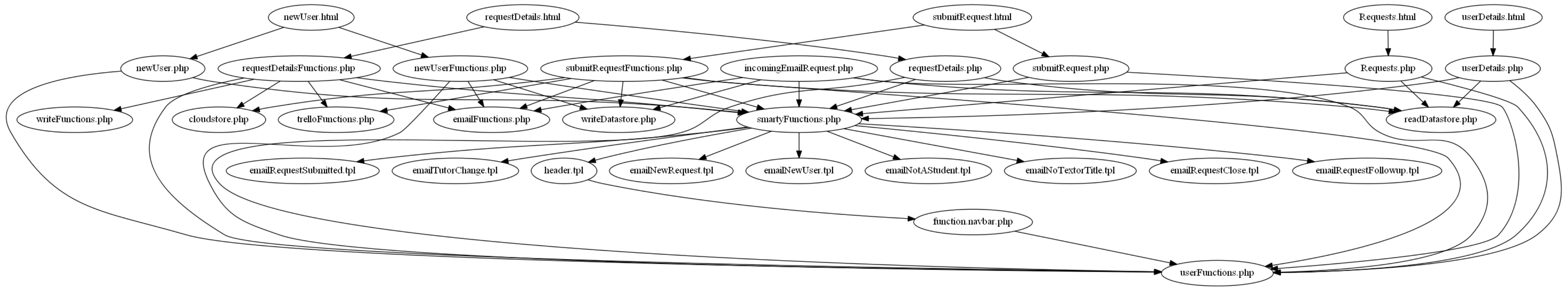
All actions would involve authentication but that is not shown to prevent the use case being more messy.

JavaScript module dependency diagram



Generated using PHPStorm.

File dependency diagram



Shows the mapping of require_once, $smarty->display and file_get_contents between files, vendor/autoload.php is not included as all php files use it and the same for footer.tpl with html files.
Generated using Graphviz.

F. Notes

Unresolved Bugs

Enabling cURL using ‘google_app_engine.enable_curl_lite = “1”’ will cause the site to take minutes to load pages, not fixed as HTTP Guzzle was used instead of cURL.

Randomly when signing out of an account then into another, it appears that the signout failed as the old account name will be present, resigning out then back in or waiting a minute then refreshing resolves this.

References/Bibliography

Google, 2018a. *Mail PHP API Overview | App Engine standard environment for PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/appengine/docs/standard/php/mail/ [Accessed 26 Dec 2018].

Google, 2018b. *Reading and Writing Files | App Engine standard environment for PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/appengine/docs/standard/php/googlestorage/ [Accessed 26 Dec 2018].

Google, 2018c. *Using Cloud Datastore with PHP | PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/php/getting-started/using-cloud-datastore [Accessed 26 Dec 2018].

Google, 2017. *API Documentation | App Engine standard environment for PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/appengine/docs/standard/php/refdocs/classes/google.appengine.api.users.UserService [Accessed 26 Dec 2018].

Google, 2018d. *User Authentication Options | App Engine standard environment for PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/appengine/docs/standard/php/oauth/#google_sign-in [Accessed 26 Dec 2018].

Maguire, S., 2018. *stevenmaguire/trello-php - Packagist* [online]. Packagist.org. Available from: https://packagist.org/packages/stevenmaguire/trello-php [Accessed 26 Dec 2018].

Google, 2018e. *Sending and Receiving Mail with the Mail API | App Engine standard environment for PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/appengine/docs/standard/php/mail/sending-receiving-with-mail-api [Accessed 26 Dec 2018].

Occhipinti, D., 2014. *floriansemm/official-library-php-email-parser - Packagist* [online]. Packagist.org. Available from: https://packagist.org/packages/floriansemm/official-library-php-email-parser [Accessed 26 Dec 2018].

Google, 2018f. *google/cloud-storage - Packagist* [online]. Packagist.org. Available from: https://packagist.org/packages/google/cloud-storage [Accessed 26 Dec 2018].

Ohrt, M., 2018. *smarty/smarty - Packagist* [online]. Packagist.org. Available from: https://packagist.org/packages/smarty/smarty [Accessed 26 Dec 2018].

Walder, T., 2018. *tomwalder/php-gds - Packagist* [online]. Packagist.org. Available from: https://packagist.org/packages/tomwalder/php-gds [Accessed 26 Dec 2018].

Google, 2018g. *google/cloud-datastore - Packagist* [online]. Packagist.org. Available from: https://packagist.org/packages/google/cloud-datastore [Accessed 26 Dec 2018].

Google, 2018h. *app.yaml Reference | App Engine standard environment for PHP | Google Cloud* [online]. Google Cloud. Available from: https://cloud.google.com/appengine/docs/standard/php/config/appref [Accessed 26 Dec 2018].