**Project README and Log - Orbital – Tay Guo Qiang & Won Jun Ru Daphne (Documentation for Milestone #3)**

**Project Title: Schedule Comparator**

Table of Contents

**Readme 2-5**

Overview2

Intended features 2

User stories2

Project status3

Features finished up to now3

Planned features after current milestone 4

Proposed level of achievement 4

References used 5

**Project Log6-8**

Time spreadsheet 6

**README**

|  |
| --- |
| **Overview**: Our project is to come up with a schedule comparison web application that helps groups of people to compare schedules and find out a common time available for a meet-up. We plan to have both the manual entry and the timetable-comparison option in our web application.    Slide for project idea  **Intended features:**   * Temporary sessions created using MySQL database with PHP, where groups of users can come together to collate the availability schedules of each person without requiring logins * Date-picker and time-picker input for users to enter schedule inputs * NUS Timetable URL upload comparison option   **User stories:**   * As a general user, I want to use this web application to collate the schedules of everyone in my group and find out which dates and times we are all available, so that we can all meet up together for projects or group activities. (Schedule-comparing functionality) * As a general user, I want to be able to modify the schedule I have input into the application to reflect any changes to my schedule so that the schedule comparison would have taken those changes into account and not affect the accuracy of the comparison. (Ability to modify entry) * As a general user, I do not wish to have an account for login just to use this application, yet I would also want to ensure that no one else besides my group members are involved in our own group’s schedule comparison. (Temporary sessions maintained by PHP-MySQL) * As an NUS student, I want to be able to use my NUSMods timetable together with my fellow NUS friends’/project mates’ timetables to find a common date where we can all meet up together, so that we can meet up for a project discussion. (NUSMods Timetable URL option)   This web application works somewhat differently from Doodle Poll, contrary to general opinion. For Doodle Poll, a date is set and the group of people indicate in Doodle Poll whether they are available on that specified date or not. However, for our Schedule Comparator, everyone’s schedules are collated together and the comparator attempts to find a common date that everyone can meet up together. If there is not a single common date where everyone can meet up, certainly a compromise is needed in the schedules.  **Project status:**   * Link to video: https://www.youtube.com/watch?v=ojZGxuiVCfk * Link to application: <http://schedulecomparator.net46.net/> * Link to Github: <https://github.com/tgqiang/schedule_comparator> * Application is now available for testing   **Features finished up to now:**   * The general application interface has been completed, but may be further polished over time. * Sessions system using PHP-MySQL has been implemented. * Manual and timetable comparison functionality is up. * Application has been deployed on a server and therefore we have removed the screenshots that were in previous READMEs.   Elaboration about our application:  The sessions system:   * When one member in the group creates a session, effectively he/she is creating a table with table name set as the input session ID in a database that has been pre-created for our comparator. * Rationale for unique session ID is because we use it for URL redirection as well as to prevent unrelated users from easily accessing other people’s sessions via an ID of ‘1’ or something similar, seeing that we do not have a true login system. * Other members in the same group that joins that particular session would be effectively accessing the same table in the database. * Inputs from both the session-creator and the other members would simply be appended to the table. * When every single person in the group is done, the common dates can be computed from the table data. * When the group no longer wishes to use this comparator, the creator can destroy the session, which effectively deletes the table so that it is rendered inaccessible anymore.   Manual comparison system:   * For this comparison, every group member will gain access to the same page (based on their group session ID). All members in the group are to fill in their schedules with the interface and append their entries to the table. * The intention of the page is to fetch the updated table each time an entry is appended to the table. It should also allow each user to edit their own entry’s schedule if there happens to be a change while in the midst of the comparison process.   Timetable comparison system:   * For this comparison, every group member will gain access to the same page (based on their group session ID). All members in the group are to input their NUSMods timetable URLs. * The intention of the page is to make use of the NUSMods API to retrieve the common available dates and times in the users’ timetables.   Application exit:   * When the group is done using the application, the members who joined the session would leave the session upon clicking the exit button whereas the one who created the session would destroy the session as well as exit the session upon clicking the exit button.   Input validation & Exception handling:   * First scenario would be SQL injection attacks, which would totally ruin the functionality of our application’s session system if SQL queries in the form of ‘DROP TABLE’ or ‘DROP DATABASE’ manage to slip into our application. Therefore, our application does not permit sensitive SQL keywords such as ‘TABLE’, ‘DROP’, ‘DATABASE’, etc. * Second scenario would be when a particular session ID is already in use. To prevent different groups from accessing the same table via the session ID, we check to see if a table of the same session ID name exists before allowing the session-creator to proceed. * There exists a third scenario whereby the creator leaves the session ahead of those members who joined the session, thus causing the session to be destroyed even before those members can respond. In this case, we currently rely on users to communicate with each other on this aspect. Since we do not have a true login system, determining the state when the session should be terminated is difficult. One solution would be to implement a chat system in the application but we are currently running out of time. Therefore, we will implement this in the future, after Orbital.   **Planned Features after current milestone:**   * We are currently in the phase of refining the user interface of our application, after receiving feedbacks about our application’s poor interface.   **Proposed level of achievement:**  We are aiming for the Gemini level of achievement for our project. We propose why we should be granted the Gemini level of achievement below:  We have used different platforms as listed below:   1. We use a WAMP stack architecture for our Schedule Comparator instead of using GAE 2. We use multiple Javascript components for our web application (i.e. PHP, Ajax, jQuery) 3. Our sessions system serve as an administrative front end for our Schedule Comparator 4. We have input validation and exception handling to guard against SQL injections (since we are using MySQL for our tables-implemented sessions system) 5. We use Git for version control, moving on from the Google Drive phase 6. We use NUSMods’ API   We are also active in our application development and our application is more or less ready.  **References Used:** *For self-learning:*   * W3Schools (<http://www.w3schools.com/>) – things for web development can be learnt here * Codecademy ([http://www.codecademy.com/)](https://developers.google.com/appengine/docs/python/%29) * Code School (<https://www.codeschool.com/>) – however to complete the full course, payment subscription is needed * Free Code Camp (<http://www.freecodecamp.com/>) * Meteor.js tutorial (<http://meteortips.com/first-meteor-tutorial/projects/>) * stackoverflow (<http://stackoverflow.com/>) – the source of answers to our programming problems   *Features that we used in our Schedule Comparator so far that we managed to find:*   * Multi Date-Picker (<http://multidatespickr.sourceforge.net/>) for the date-picker feature we initially used but dropped in the end due to application changes * Time-picker (<https://github.com/fgelinas/timepicker>) for the time-picker feature used * Bootstrap (<http://getbootstrap.com/>) for the Bootstrap we used * jQuery (<https://jquery.com/>) for the jQuery used. Information about jQuery UI can be accessed from here too. * PHP (<https://secure.php.net/>) for the PHP used * MySQL (<https://www.mysql.com/>) for MySQL used * MomentJS (<http://momentjs.com/>), a Javascript library for date manipulations * moment-range (<https://github.com/gf3/moment-range>), an extension of MomentJS * Bitnami (<https://bitnami.com/>), where we found our WAMP stack architecture from. It also offers many other useful stuff. * Hopscotch (<https://github.com/linkedin/hopscotch>) for the application tutorial used * GitHub (<https://github.com>) for version control employed. See project status for our GitHub repository link. |

**Project Log**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The application is now completed.  Specified here are the accumulated time figures. For more in-depth elaboration of how we used these hours, please see the spreadsheet below.  Accumulated hours for Guo Qiang: 164  Accumulated hours for Daphne: 139  Total hours together: 303   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **S/N** | **What** | **Date** | **Guo Qiang (Duration in hours)** | **Remarks** | **Daphne (Duration in hours)** | **Remarks** | | 1 | Liftoff Day 1 | 11/5/2015 | 8 | - | 8 | - | | 2 | Liftoff Day 2 | 12/5/2015 | 8 | - | 8 | - | | 3 | Self-learning (HTML/CSS/Bootstrap/jQuery) + Coding for HTML, CSS, Bootstrap & jQuery | 14/5/2015 to 22/5/2015 | 30 | Coded the base idea for the web application (the starting html, web interactivity and design). | 30 | Self-learning from Codecademy (HTML/CSS/Javascript /Python) | | 4 | Mission Control: MeteorJS | 27/5/2015 | 2 | - | - | - | | 5 | Self-learning (Ruby) | 30/5/2015 to 31/5/2015 | 5 | - | - | - | | 6 | Self-learning (SQL) | 31/5/2015 to 5/6/2015 | 7 | - | - | - | | 7 | Self-learning (Git) | 31/5/2015 to 2/6/2015 | 3 | Some basics of Git have been learnt. Learning phase is still incomplete though. | - | - | | 8 | Self-learning (jQuery) | 1/6/2015 to 23/6/2015 | - | - | 12 | - | | 9 | Self-learning (PHP) | 5/6/2015 to 21/6/2015 | - | - | 10 | - | | 10 | Self-learning (PHP) | 7/6/2015 | 2 | - | - | - | | 11 | Self-learning (Ruby) | 10/6/2015 to 13/6/2015 | - | - | 12 | - | | 12 | Website coding for server-side scripting (for PHP-MySQL functionality) | 18/6/2015 to 28/6/2015 | 33 | See next ‘Remarks’ | - | Difficulties with PHP and its installation/configuration was faced here, thus resulting in much time spent in this phase | | 13 | Meet-up 1 | 24/6/2015 | 6 | See next ‘Remarks’ | 6 | To debug PHP issues | | 14 | Mission Control: Android App | 24/6/2015 | - | - | 2 | - | | 15 | Edit design for webpage | 25/6/2015 | - | - | 3 | - | | 16 | Meet-up 2 | 28/6/2015 | 5 | See next ‘Remarks’ | 5 | For milestone 2 | | 17 | Implementing PHP for adding entry, updating entry, refreshing table and date-time computation | 30/6/2015 | 6 | - | - | - | | 18 | Coding functions to be used for timetable-comparison option in trial mode | 5/7/2015 | 6 | Trial mode to check if functions work as intended. | - | - | | 19 | Integrating previously coded functions into timetable-comparison option | 7/7/2015 | 3 | Functions work and are officially added into the application page | - | - | | 20 | Debugging integration error in (19) | 8/7/2015 | 3 | Ran into some problems when trying to integrate (18) into application, thus debugging needed. | - | - | | 21 | Completion of timetable-comparison option, improved help dialog and code tidy-up | 9/7/2015 | 5 | - | - | - | | 22 | Integration of date-time feature in manual-comparison option | 12/7/2015 | 4 | Changing functionality of manual option: from dates to date-time | - | - | | 23 | Testing #1 | 13/7/2015 | - | - | 12 | Testing application for bugs | | 24 | Debugging after round 1 testing | 14/7/2015 | 8 | Testing and fixing of session creation functionality | - | - | | 25 | Integrating new interface for manual-comparison option | 15/7/2015 | 6 | Revised interface for receiving date-time inputs | - | - | | 26 | Revising computation function for manual-comparison option to account for date-time inputs and integrating MomentJS and moment-range | 16/7/2015 | 6 | Revised computation function to adjust for date-time inputs | - | - | | 27 | Testing #2 | 18/7/2015 | - | - | 6 | - | | 28 | Debugging for round 2 testing | 19/7/2015 to 20/7/2015 | - | - | 9 | Session-join functionality fails to work as intended, thus debugging needed. | | 29 | Meet-up 3 | 21/7/2015 | 4 | Deployment of application and further bug-fixes (for session-join functionality) | 4 | Design changes, refining tutorial and UI improvements | | 30 | Design + UI revamp, integration of application tour using Hopscotch JS | 22/7/2015 to 23/7/2015 | - | - | 8 | - | | 31 | Tutorial page revamp | 28/7/2015 | 3 | Making pictorials and modifying the help page for users’ ease of understanding how to use our application | - | - | | 32 | Debugging application tour | 30/7/2015 | - | - | 4 | Application tour failed to load, thus debugging needed. | |  | **Accumulated Hours** |  | **164** |  | **139** | **Total: 303** | |