# Small group project self-assessment

**Team name**: *Hyena*

**URL**: https://Jakerpppp.pythonanywhere.com

The table below contains a list of epics mentioned in the assignment. For each epic that the team produced features for, add a brief explanation outlining how the user can access and use it. You have been provided with some code that incorporates some of the features you need (though you can extend it). This self-assessment will be used to help the marker figure out how to access your system, so try to keep it clear and concise.

Do review the marking scheme for the small group project to understand how the project is assessed. You are reminded that the assignment is **not** to try and implement every single epic.

|  |  |
| --- | --- |
| **Epic** | **Implementation** |
| Epic 1  The web application should have user authentication and authorisation mechanisms. Users should be able to create accounts, log in, and manage their profiles. | *The features in this cell were provided in the scaffolding project.*   * *A home screen gives users the option to sign-up or login (URL/)* * *Users without account, who are not logged in, can sign up as new users (URL/sign\_up)* * *Users with an account can log in (URL/log\_in)* * *Logged in users can log out (URL/log\_out)* * *Logged in users can change their password (URL/password)* * *Logged in users can change their profile (URL/profile)* * *After completing an account related task, logged in users are redirected to a currently blank dashboard (URL/dashboard)* * *If a not-logged in user tries to access pages requiring login, they are redirected to the login page, and then to the page they were after following successful login.* * *A menu is available on all pages while users are logged in.* |
| No further implementation |
| Epic 2  Users can form teams, and invite team members to their team.  Further implementation:  Edit the Team details and delete teams.  Only Team Members can access their team. | * *Logged in Users can create a team (URL/create\_team)* * *Logged in Users can access their team via (URL/team/team\_id) where team\_id is the primary key for the team* * *Logged In Users can invite team members if they know the other persons email e.g* [*jane.doe@example.org*](mailto:jane.doe@example.org) *(URL/team/team\_id/invite)* * *The creator of the team can edit the team with (URL/team/edit\_team/team\_id)* * *The creator of the team can edit the team with (URL/team/edit\_team/team\_id)* |
| Epic 3  Each user can create tasks, assign tasks to other team members, and set due dates  Further Implementation:  Edit tasks and delete tasks.  Archive and Unarchive Tasks.  Mark Tasks Completed. | * *Users that have a team can create a task for that team, and set a due date while creating the task (URL/create\_task/team\_id)* * *Users that have a team can view the tasks for that team via (URL/team/team\_id) where team\_id is the primary key for the team* * *Users that have a team can assign tasks to other members of the team (URL/* *assign\_member\_to\_task/task\_id/user\_id) where user\_id is the primary key for the user to be assigned* * *Users that have a team can edit the tasks (URL/edit\_task/task\_id) where task\_id is the primary for the task.* * *The author of a team can delete tasks in a team (URL/* *delete\_task/task\_id)* * *Users of a team can toggle if a task is completed (URL/task\_toggle/task\_id)* * *Users can toggle archiving a task instead of deleting it for reference(toggle\_archive/task\_id)* |
| Epic 4  Users provide a dashboard to view and manage assigned tasks, and team tasks.  Further Implementation:  Dashboard displays overdue tasks | * *Logged in Users are redirected to the dashboard, this displays all the tasks assigned to them, and all of their teams. (URL/dashboard)* * *Logged in Users can click on a task to be taken to the team page for that task, so they can manage their tasks. (URL/team/team\_id)* * *Logged in Users can click on a team to manage that team (URL/team/team\_id)* |
| Epic 5  Enable users to search, order and filter tasks. These could be based on name, completion status, priority, due date ranges, assigned developer, or team/project. | Not implemented |
| Epic 6  Add a priority system for tasks, allowing users to assign priority levels and filter tasks based on priority. Include automated reminders for high-priority tasks nearing their due dates. | Not implemented |
| Epic 7  Introduce task dependencies, enabling users to link tasks and establish dependencies between them. Ensure that the application adjusts task timelines based on dependencies. | Not implemented |
| Epic 8  Integrate a time tracking feature for tasks, allowing users to log time spent on each task. Provide summary reports of time spent on tasks over different periods. | Not implemented |
| Epic 9  Implement an activity log that records user actions within the application, providing an audit trail for accountability. | Not implemented |
| Epic 10  Add a notification system (within the constraints of a Django) to alert users about task assignments and upcoming due dates. | Not implemented |
| Epic 11  Introduce gamification elements to motivate users, such as achievement badges, leaderboards, or a points system based on task completion. | Not implemented |