# Small group project self-assessment

**Team name**: Penguin

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. | * A button on the Task Manager bar gives users the option to create a new team (URL/create\_team). * A team must have a unique name. * Users can add other users to their teams. * Once the team is created, user is redirected back to the dashboard. (URL/dashboard) and a message that tells the user that team was created successfully is displayed. * If a user entered invalid team name they are redirected back to the same page (URL/create\_team). * Users can see what teams they are in (URL/teams). From here they can click a button to create a new team or to edit one of their teams * Users are allowed to edit teams (URL/edit\_team). To do so they must enter a current team name. They can change the name of the team, add members or delete members. * If a user entered current team name that does not exist or a new team name is not unique they are redirected back to the same page (URL/edit\_team) and corresponding errors are displayed * Once the team is edited, user is redirected back to the dashboard. (URL/dashboard) and a message that tells the user that team was created successfully is displayed. |
| Epic 3  Each user can create tasks, assign tasks to other team members, and set due dates. | * A button on the Task Manager bar gives users the option to create a new task (URL/task/create). * Users can assign tasks to other users on the same page. They also have to input the title, description and due date of the task. * Users are not allowed to input a due date in the past. * Once the task is created, users are redirected to a new page to see all the tasks assigned to all the users. (URL/ task/all) * Users can delete the task if needed (URL/ task/delete). The user is not redirected to a new page. * Users can manage the task and change any details needed. Once the users clicks the Manage Task button, they are redirected to the create task page, but the page is populated with the information of the task. (URL/ task/create). |
| Epic 4  Users provide a dashboard to view and manage assigned tasks, and team tasks. | Not implemented |
| 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. | * User can see the activity log in the terminal. * It also shows the activity log for user that do not log into account. * The activity log provides the URLs, which user has visited. * If user provides input, that uses database, the activity log shows the URL with POST method. * If user do not provide input, the activity log shows the URL with GET method. |
| 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 |