# Small group project self-assessment

**Team name**: *Lemming*

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. | * On the dashboard after logging in, a user can create a team. They are then redirected to the team page. * The team page shows a list of the users in the team. * On the team page, the creator of the team (the admin) can add members via a search bar. * To add a team member, the admin can search using the first name, last name or username. * They can also remove users from the team with the Remove Member Button. |
| Epic 3  Each user can create tasks, assign tasks to other team members, and set due dates. | * Users who are not part of a team are prompted to create a team on the dashboard. * After creating a team, the create task card appears on the dashboard. * The user can then create new tasks, which involve setting the due date of the task. * The user can edit the task later to assign or remove team members and edit other details. |
| Epic 4  Users provide a dashboard to view and manage assigned tasks, and team tasks. | * The dashboard shows all tasks created for the current team as cards. * A sidebar on the left allows the user to show the dashboard for each team they are part of. * The user can click on each task card to view and edit the task details. * There is also a button to view the team details for the current team. * The dashboard also shows a list of due dates of all the tasks the user has access to. |
| 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. | * Users can search for tasks that are on their dashboard with a search bar. * Users can filter tasks using a dropdown on the dashboard based on completion and levels of priority. * Users can order tasks by various properties such as completion status, priority, name, and due date, selected through a drop down. |
| 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. | * Tasks are assigned a priority on creation, ranging from low to high. This can be edited. * High priority tasks have a warning on them on the dashboard. * Notifications for tasks are sent when the task’s due date is nearing. * By default, the reminder will be sent 0 days before, but this can be edited in the edit task page. * They can be viewed in the notification popup at the top of the screen. |
| 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. | * Users can log the time they have spent on each task by manually inputting it on a task’s page. * A summary report can be viewed on the dashboard. This shows the time spent on each task as well as how much time the team has spent working on tasks per day. * The user can clear the time they’ve spent on a task on the task edit page. They can also clear the time tracking for a task completely. |
| Epic 9  Implement an activity log that records user actions within the application, providing an audit trail for accountability. | * The server keeps track of significant actions each user makes, such as logging in, creating a task etc. * The admin of a team can view each team members’ actions on the team page by clicking on the view activity log button next to each member. * This shows the user’s activity log in the form of a table. |
| 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 |