# Small group project self-assessment

**Team name**: *Newt*

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. | * Users can form teams * Search for teams * Request to join teams * Search other users and invite them to teams * Delete the team if they are owner * Leave the team |
| Epic 3  Each user can create tasks, assign tasks to other team members, and set due dates. | * User can assign a task to any of their teams or themselves * User can assign task to any member of their team or none of them but has to have selected team first |
| Epic 4  Users provide a dashboard to view and manage assigned tasks, and team tasks. | * Main dashboard shows all tasks owned by user or assigned to user * User can edit task details by pressing edit button if they own the task |
| 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 teams, other users and tasks in search bar in navbar * Filtering of tasks based on priority, and completion state * Ordering of tasks based on creation date, priority or state of task |
| 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. | * Priority system with 3 levels of priority * No automated reminder system |
| 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 |