*This schedule offers a summarized overview of each deliverable.   
See the deliverables document for the full requirements of each deliverable.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Week** | **Deliverable** | **Requirements** | **Demo** |
| 1/14 | 0 | Semester start | | |
| 1/21 | 1 | Project Kick-off | * Each team member should start to create Slack and GitHub accounts |  |
| 1/28 | 2 | Deliverable 0  Project Setup | * Team has sent mentors list of GitHub usernames via Slack * Team’s copy of template repository set up * Each team member has cloned template project and can run it on local machine using instructions in README (GCP does NOT need to be set up yet) |  |
| 2/4 | 3 | Deliverable 1  ERD | * Entity Relationship Diagram describing your database tables |  |
| 2/11 | 4 | **Milestone 1** | * Team repository is set up (clone of template project) * Pages are scaffolded out with routing to access them all (pages can be mostly empty, like the home page just saying “Home”, etc.) * Call example Java endpoint (/ping) from client and display the results in the frontend (ask mentors for more details about this in our scrum meetings) | * Demonstrate that all pages exist by directly hitting their URLs (/home, /profile, etc.) * Display the result of the test endpoint service call |
| 2/18 | 5 | Deliverable 2  Account Creation & Login | * User account creation and login | * Demonstrate creating a user and logging in with that user |
| 2/25 | 6 | Deliverable 3  E2E Service Demo | * At least one full end-to-end (E2E) service is complete | * Demonstrate that an action in the UI will make a request to the backend, requesting data from the MySQL database, and the response will be passed back to the frontend and displayed to the user |
| 3/3 | 7 | Spring break | | |
| 3/10 | 8 | **Milestone 2** | * Every page is implemented – sign-in/registration, profile page, home page, meetup creation/editing (at a minimum) * In the service layer, be hitting a MySQL database and be able to create/edit account, add/edit classes/areas of study, add/edit “study buddies”, add/edit profile details, create/edit meetups, review tutors, etc. * Project is deployed to GCP * Continuous integration is set up (a Git commit triggers a build, which triggers a GCP deployment) * In the frontend, an individual should be able to: * Register and log in * Select user type (student vs tutor) * Add and edit classes/areas of study * Search for and mark other users as “study buddies” * Create and edit study meetups * Add unit tests to the Java (API) layer and add a testing step to the CI/CD pipeline (teams should maintain, update, and expand these tests going forward) | * Show the app running live on GCP * Change some text on the home page and commit the change, then show that the CI/CD pipeline is working and that the message changed on the live site * Demonstrate creating a new account and logging in * Demonstrate creating a new study meetup and providing its details (date, time, location, subject, etc.) * Show the deployment pipeline passing in GitHub Actions, as well as showing that unit tests were run and are passing |
| 3/17 | 9 | *No deliverable* |  |  |
| 3/24 | 10 | Deliverable 4  Recommendations | * Progress with the recommendations system to both find users and meetups * Recommendations should use criteria such as subjects, study methods, type of study partners (students or tutors) | * Demonstrate recommendations for a user based on small set of criteria |
| 3/31 | 11 | Deliverable 5  Notifications | * Progress with the notifications system – viewing messages, marking them as read, deleting * App should notify user if there are unread messages | * Demonstrate notifications for a user when another tries to connect with them or one of their meetups * Demonstrate notification for upcoming meetup |
| 4/7 | 12 | **Milestone 3** | * Recommendations: A critical component of this project is a sophisticated system for connecting users based on shared interests. Typically, teams achieve this by allowing a user to enter classes/areas of study they wish to study, select whether they are searching for students or tutors, and set preferences for study methods. Then, the system can recommend a list of other users that the user can become buddies with or schedule meetups with. You can take a different approach to this one and even add components like suggesting study playlists or locations, but your recommendations engine needs to be capable of finding relevant students to create a meetup with and support a large user base. * Notifications: Your system should provide notifications to its users and alert them when they have unread notifications. At a minimum, users should receive notifications when other users become buddies with them, when users join one of their meetups, and when a meetup that they have joined is about to start. * In the service layer, be able to:   + Suggest other users to become buddies with   + Suggest other students and tutors to add to a meetup/create a meetup with   + Schedule, edit, and cancel meetups   + Send notifications   + Submit ratings/reviews for tutors   + Update existing and add new unit tests in the Java layer * In the front-end, individuals should be able to:   + Review recommended students and tutors   + Become buddies with and join/schedule meetups with recommended students and tutors   + Easily allow one’s study buddies to join a meetup they own   + Get notifications about study buddy and meetup statuses   + Edit or cancel a meetup   + Rate the performance of a tutor after a tutoring session | * Demonstrate suggestions for students and tutors to connect with, related to study preferences and subject * Demonstrate that the system sends a reminder notification before a meetup that the user has joined * Demonstrate rating a tutor after a meetup |
| 4/14 | 13 | Deliverable 6  UI Hardening | *Note: This is the last deliverable expected for the project until your final presentations* | * As the project nears completion, we want to see how you’ve changed and finalized your UI in preparation for the final presentation |
| 4/21 | 14 | *No deliverable* |  |  |
| 4/28 | 15 | **Milestone 4**  (Due at final presentations) | * Security features   + Input validation   + User authentication (do unauthenticated users receive a different view of the site? etc.)   + User authorization (for example, a validated user can only edit their own classes/areas of study, but can see others’ in a read-only format)   + Password encryption * Final UI/UX design * Bonus features | *The final demo should show off your app completely. However, this is the list of things specifically from Milestone 4 that we would like to see:*   * Demonstrate security * Demonstrate bonus features * The product and UI should be in a complete state |