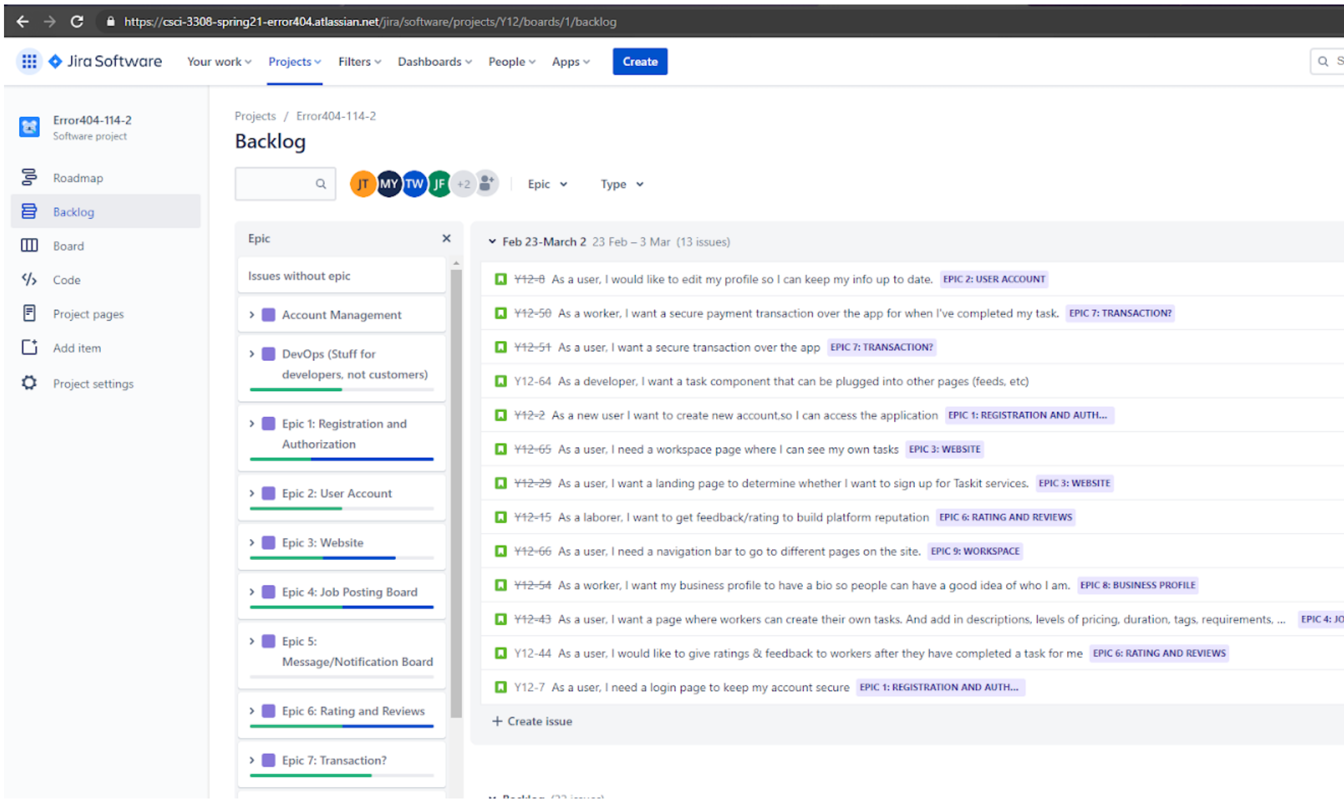
- **Title:** Taskit
- **Team Members:**
    - Gregory Tanaka | Github: TheGregTanaka
    - Josh Truong | Github: srirachaBoi -> jtdev21
    - Manoj Yeddanapudy | Github: maye7573
    - Jules Fischer-White | Github: julesiam
    - Timothy Waymouth | Github: timway33
    - Ryan Osler | Github: GitOsler
- **Project Description**
    - At Taskit enterprise, we have rethought short-term contracting. Ever had a small job and needed another pair of hands? We'll connect you to verified locals who would love to help. We also provide a safe payment service and a marketplace for you to advertise your skills.
- **Project Tracker:** (Trello, Asana, or similar tool.)
    - Link to Project Tracker



Epic
- Issues without epic
- Account Management
- DevOps (Stuff for developers, not customers)
- Epic 1: Registration and Authorization
- Epic 2: User Account
- Epic 3: Website
- Epic 4: Job Posting Board
- Epic 5: Message/Notification Board
- Epic 6: Rating and Reviews
- Epic 7: Transaction?

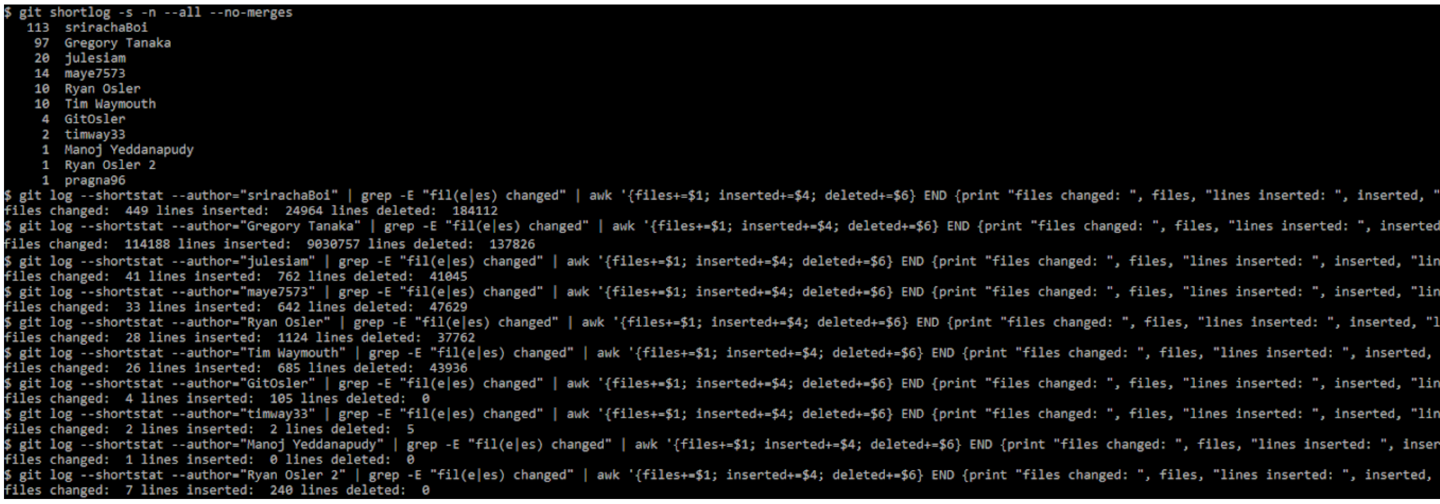
Feb 23-March 2 23 Feb – 3 Mar (13 issues)

- ~~Y12-8~~ As a user, I would like to edit my profile so I can keep my info up to date. EPIC 2: USER ACCOUNT
- ~~Y12-50~~ As a worker, I want a secure payment transaction over the app for when I've completed my task. EPIC 7: TRANSACTION?
- ~~Y12-51~~ As a user, I want a secure transaction over the app EPIC 7: TRANSACTION?
- Y12-64 As a developer, I want a task component that can be plugged into other pages (feeds, etc)
- ~~Y12-2~~ As a new user I want to create new account,so I can access the application EPIC 1: REGISTRATION AND AUTH...
- ~~Y12-65~~ As a user, I need a workspace page where I can see my own tasks EPIC 3: WEBSITE
- ~~Y12-29~~ As a user, I want a landing page to determine whether I want to sign up for Taskit services. EPIC 3: WEBSITE
- ~~Y12-15~~ As a laborer, I want to get feedback/rating to build platform reputation EPIC 6: RATING AND REVIEWS
- ~~Y12-66~~ As a user, I need a navigation bar to go to different pages on the site. EPIC 9: WORKSPACE
- ~~Y12-54~~ As a worker, I want my business profile to have a bio so people can have a good idea of who I am. EPIC 8: BUSINESS PROFILE
- ~~Y12-43~~ As a user, I want a page where workers can create their own tasks. And add in descriptions, levels of pricing, duration, tags, requirements, ... EPIC 4: JO
- Y12-44 As a user, I would like to give ratings & feedback to workers after they have completed a task for me EPIC 6: RATING AND REVIEWS
- Y12-7 As a user, I need a login page to keep my account secure EPIC 1: REGISTRATION AND AUTH...

+ Create issue

-
- VCS: Link to your git Repository
    - Github
    - - Test Cases
    - - README.md in GitHub explaining to others what your project is about and how to run your project code.
    - - Project Milestone 7 document titled as Project Milestone7_[TeamNumber].pdf

Contributions



```
$ git shortlog -s -n --all --no-merges
   113  srirachaBoi
    97  Gregory Tanaka
    20  julesiam
    14  maye7573
    10  Ryan Osler
    10  Tim Waymouth
     4  GitOsler
     2  timway33
     1  Manoj Yeddanapudy
     1  Ryan Osler 2
     1  pragna96
$ git log --shortstat --author="srirachaBoi" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted, "
files changed:  449 lines inserted:  24964 lines deleted:  184112
$ git log --shortstat --author="Gregory Tanaka" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted
files changed:  114188 lines inserted:  9030757 lines deleted:  137826
$ git log --shortstat --author="julesiam" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted, "lin
files changed:  41 lines inserted:  762 lines deleted:  41045
$ git log --shortstat --author="maye7573" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted, "lin
files changed:  33 lines inserted:  642 lines deleted:  47629
$ git log --shortstat --author="Ryan Osler" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted, "l
files changed:  28 lines inserted:  1124 lines deleted:  37762
$ git log --shortstat --author="Tim Waymouth" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted,
files changed:  26 lines inserted:  685 lines deleted:  43936
$ git log --shortstat --author="GitOsler" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted, "lin
files changed:  4 lines inserted:  105 lines deleted:  0
$ git log --shortstat --author="timway33" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted, "lin
files changed:  2 lines inserted:  2 lines deleted:  5
$ git log --shortstat --author="Manoj Yeddanapudy" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inser
files changed:  1 lines inserted:  0 lines deleted:  0
$ git log --shortstat --author="Ryan Osler 2" | grep -E "fil(e|es) changed" | awk '{files+=$1; inserted+=$4; deleted+=$6} END {print "files changed: ", files, "lines inserted: ", inserted,
files changed:  7 lines inserted:  240 lines deleted:  0
```

- Joshua Truong
  - Worked with react js, node js(express), MySQL & workbench, insomnia, Stripe API, Axios, material UI, socket.io(dropped chat feature),
  - Contributed towards the making of our database
  - Contributed towards the making of the project wireframe: page_1 | page_2
  - Made the architecture diagram
  - Worked on Tasks component/feature (front and back)
    - Handles displaying information in either a minimized or enlarged window. Displays task name, address, price, description, contact information(email & phone), task status
      - Worked on making sure tasks can be dropped, completed, deleted, and created.
    - There are 4 types of task-based on task status:
      - Accepted, created, required confirmations, and payment tasks
    - Display Accepted Task feature
      - 2 options allow a worker to complete tasks or drop tasks if they cannot be completed
    - Display Created Task feature
      - Task created by the tasker will display a task status
      - The tasker has the option to delete a task but only if a task has been completed or a task is still in pending status
    - Display Required Confirmation of a task
      - The user will be able to confirm a completed task by writing a review to the worker.
    - Display Required Payment task
      - Tasker will be able to pay workers by clicking on the flickering card notification and adding billing address as well as card information.

    - Worked on Payment feature(front and back)
        - Used Stripe API (functional only on testing mode and not live).
            - **Transaction process:** tasker pays the specified amount -> our platform takes a 2.13% cut -> the worker receives the rest of the transaction.
    - Worked on Workspace(front and back)
        - Worked on displaying a list of categorized task (accepted, created, required confirmation, required payment) in accordions
        - Display button option for tasker/worker to view on their company profile
        - Allow users to create a task
    - Worked on Company Profile(front and back)
        - Worked on displaying worker/company profile (completed tasks, reviews(rating, reviewer name, review), bio, contact info, average rating of worker)
    - Worked on Creating a review and display review(front and back)
        - Creating a review has 2 parts rating a worker from 1-5 and writing a description
    - Worked on allowing users to create a task(front and back)
        - Allows a user to create a task by entering required fields such as task name, price, type, address, and description. Automatically the status of a task is set to pending and displayed on the exploring/feed page
    - Worked on Chat Feature (front)
        - Feature dropped due to time constraints. You can view the chat UI by clicking here.
    - Worked on Landing Page (front)
        - Collab with Ryan Osler to convert Ryan's pure HTML and CSS to react component.
- Manoj Yeddanapudy
    - Worked on Registration Page(worked on both frontend and backend)
        - Front-End: was made in react.js using Material UI for components and features
            - Biggest component used was <TextField> which allows users to enter their information. There was 5 text fields
            - Simply to allow users to enter their information and click submit
            - Redirect on the bottom of the page if you are already registered
        - Back-End: was done in node.js
            - Requires the user to enter all five fields, when user clicks submit a "post" request is run and the user is sent to the database to see if it exists. If it exists user is told
            - If a user doesn't exist then they are added to the database and stored with their password hashed.Then they are redirected to login page
    - Worked on Login Page(only frontend)
        - Frontend: react.js

        - Just switched the layout of our login page to a similar design to the registration page.
        - Again used Material-UI in order to accomplish this
        - **Backend down by Greg**
        -
    - Worked on Salt & Hashing passwords
        - Worked on a function that would take the password field and run a hash algorithm on the plain text then using salts it radmoizes it. All of this was achieved using bcrypt.
        - Resource: bcrypt
    - Worked on cookie function (not implemented)
        - When a user registers along with storing their information and hashing their passwords we also create a cookie that is set to them. So after registration is successful we simply pass the cookie back to log the user in immediately.
        - The refresh time of this cookie is extended to refresh after an hour.
        - Function on line 179: createCookie
- Gregory Tanaka
    - Helped initialize react and node apps in repo (npm)
    - Helped write script to structure and seed db (**MySQL**)
    - Contributed documentation to readmes (markdown)
    - **Node** (I mainly focused on backend)
        - routes for task/user CRUD operations (express framework)
        - Login functionality on the backend (JWT token, cookie validation)
    - **React**
        - Map component
        - Worked with teammates with some of the axios calls to plug backend into frontend
    - Heroku deployment

- Timothy Waymouth
    - Worked with react js, node js(express), MySQL & workbench, insomnia, Axios, material UI
    - Created the wireframe on figma to show user flow and app structure
    - Created a reusable component for the tasks using react
    - Created a feed that made calls to the backend to get the tasks that were not yet accepted and then display them
        - Also created a task view with more information on this page and made the functionality where users could accept the tasks and this would update our database to reflect the task had been accepted
        - Wrote queries in our backend to get the tasks and worked with greg to create one to update the tasks info to reflect that they had been accepted
    - Helped greg with state management across our pages to set the users to logged in after they logged in

- Jules Fischer-White
    - Designed and created project logo, both in the full word form and as a thumbnail.
    - Edit Profile Page
        - Integrated with website
        - Front end development
        - Worked with Greg and Manoj to connect to back-end
            - Post and replace edited information onto back-end
    - View Profile Page
        - Integrated with website
        - Built front end
        - Connect to back-end so user's profile information populated profile fields
    - Wrote project mission statement and description
    - Struggled to learn react, node.js, material-UI, and express and how these components work together since I had no web-development experience before this class. If we had to do it again I feel confident that I could contribute much more with the knowledge I have now.
- Ryan Osler
    - Front end design for home/landing page mostly in CSS and HTML.
    - Initially prototyped my own but then used Tim's figma wireframe as starting point
    - Navbar
        - Easy, interactive navbar tool for navigation to pages of our web app
        - Button styling and media screens (hover, color, transitions, margins)
        - Contents adjust to fit users browser size
        - Mobile style drop down menu with navbar
    - Body
        - Additional div space
        - Team members added tasks images/bootstrap cards
    - Footer
        - Company logo button to send user back to home page
        - Links to our contact info, social media, github
        - Added logos for easy read
    - Hadn't worked with a lot of these languages in this much depth until taking this course. I sacrificed a lot of time learning CSS and HTML but struggled more with javascript, node JS and react.js. Josh and Greg helped me convert a lot of my HTML into React and put it on our local server. Looking back, if I was smarter with my time I would have been able to convert to react myself and added more front-end components.

Deployment

- Frontend
    - To log in as a user who already has tasks created and assigned, log in with `gregory.tanaka@colorado.edu` PW: `Password123`
    - New users can be created through the SignUp page

        - Feed (Explore) can be viewed without logging in, however interacting with a task requires log in
        - Viewing the workspace or profile requires sign in
        - New tasks can be created from the workspace
        - Tasks can be accepted from the feed (Explore)
    - Backend
        - Routes are listed here: https://github.com/CSCI-3308-CU-Boulder/3308SP21_011_6/blob/main/code/api/README.md
- Be sure to:
    - Final Submission tag: https://github.com/CSCI-3308-CU-Boulder/3308SP21_011_6/releases/tag/v1.0
    - Include a README in your repository:
    - Describe repo organization/structure (I've outlined directories and files likely to be of primary interest to graders. For brevity, I have not listed in the repo, but this should give a clear idea of the structure.)

```
|--3308SP21_011_6 (repo root)
        |--code (contains 2 applications)
                |--api (node backend)
                        |--config
                        |--models (represents, more or less, entities from the db)
                        |--public (resources, such as images)
                        |--routes (top level route represents entity/purpose)
                        --app.js (entrypoint, routes files registered here)
                        --db.js
                        --middleware.js
                        --README.md (documentation of api routes)
                |--web (react frontend)
                        |--src
                                |--components (contains various components for the react app)
        |--data
                --database.init.sql (script to create db)
                --README.md (documentation on database structure)
        |--meetingLogs
        |--mileStones
        ---README.md (general description of project)
```

Building and running code locally

Both the front end and back end app are set up through npm, so running the application is fairly straightforward, although it does expect a local copy of the database. Below is the step by step.

1. Clone repo
2. Set up a local copy of the mysql database. The database script includes a user and password for use with development (same credentials as defined in the example.env files). This is for the sake of easy quick setup locally, however this is used on the heroku database.
3. Create .env files for both apps. Again, examples are provided with insecure credentials, different values are used in production.
4. Install dependencies for the back end
5. Build and start backend; it runs on localhost:3200
6. Install dependencies for the front end
7. Build and start front end; it runs on localhost:3000

```
# 1 clone repo
git clone git@github.com:CSCI-3308-CU-Boulder/3308SP21_011_6.git
cd 3308SP21_011_6
# 2 setup database
mysql -u root -p < data/database.init.sql

# 3 env files
cp code/web/example.env code/web/.env
cp code/api/example.env code/api/.env

# 4
cd code/api
npm install
# 5
npm start

# 6
cd ../web
npm install
# 7
npm start
```

We did not use docker or continuous integration