Project Name: ISSUE TRACKING TOOL

1. **Source code:**
    - Repository: https://github.com/rinkeshpatel22/Mean-stack-app2-issue-tracking-tool
    - git: https://github.com/rinkeshpatel22/Mean-stack-app2-issue-tracking-tool.git

2. **Deployed url:**
    - Front-end: http://rinkesh.s3-website.ap-south-1.amazonaws.com
    - Back-end API docs: http://rinkesh.co.nf/IssueTrackingToolAPIdocs

3. **Overview:** An application that allows us to create issues, update and track issues. We can manage issues by assigning to other users, set priority and status etc.

4. **Technologies used:**
    - Front-end:- Angular 8, HTML, CSS, JS, ES6, Bootstrap, Angular Material.
    - Back-end:- NodeJS, ExpressJS, Socket.io
    - Database: MongoDB

5. **Features of the application:**
    - Users can register themselves.
    - Dashboard:
        - On Dashboard, user can create a new issue.
        - User can see the list of issues with status and priorities.
        - User can sort the issue list.
        - User can search issue from the list.
        - User can filter by selection filter tag.
        - User can export the filtered list of issues in excel file.
        - User can view the update notification of issues
        - On click of any notification, user can view description of that issue
        - On click of any issue, user can view description of that issue
    - Description:
        - On Description view user can view all details of the issue.
        - User can add comment.
        - User can update the issue.
        - By clicking “Watch” button user will get update notifications of that issue.
    - Notifications:
        - Reporter, Assignee and watcher of issue will get notification if that issue get updated by any user.
        - If user in online than a notification popup will be displayed

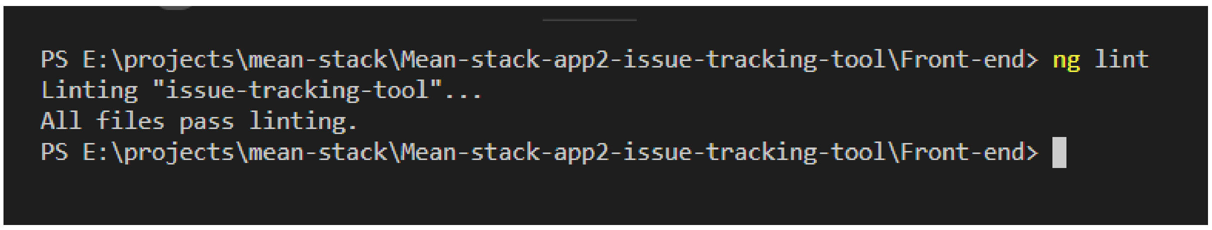
        - On click of view from popup modal user will be redirected to the description page of that issue.
        - Notifications are also sent to user’s registered email.

6. **Main modules of the application:**
    - User management
        - Signup
        - Login
        - Forgot password
        - Reset password
    - Issue management
        - Create
        - Update
        - Delete
        - Read
    - File management (attachment files of issue)
        - Upload
        - Read
        - Delete
    - Notification management (attachment files of issue)
        - Create
        - Read
    - Email management
        - Send welcome email on signup.
        - Send password reset link when requested
        - Send password change notification
        - Send Issue create notification.
        - Send Issue update notification.
    - Socket.io management
        - Establish socket connection between front-end and back-end
        - On issue update send online notification to the user who is watcher, reporter or assignee of that issue.
    - Authorization
        - Create authToken in backend and send it in successful login response to front-end.
        - Front-end pass authToken in every authorized API call
        - API verify the authToken before responding the request, if authToken is correct and not expired then response successfully otherwise reject the request.

7. **Front-end application modules:**
   - **Root Module** (app module)
   - **Core Module** - services, constants, interfaces
   - **Feature Modules**
     - **Dashboard Module** - manage dashboard page
     - **Description Module** - manage issue description page
     - **Authentication Module** - signup, login, forgot/reset password
   - **Shared Module** - header, issue-form, libraries, loader, notification
8. **Technical concepts used in front-end**
   - Lazy loading modules
   - Separation of modules: Core, Shared and Feature modules.
   - Interceptors
     - Error-interceptor: Centralized all API call error handling.
     - Token-interceptor: Passing authToken to every API call request from one common service.
     - Loader-interceptor: Centralized show/hide loading spinner on waiting for all http calls
   - Parallel API calls - forkJoin() method
   - Socket.io
   - Cookie storage
   - Responsive web design.
   - Use of interfaces and kept hard coded strings in constants files.

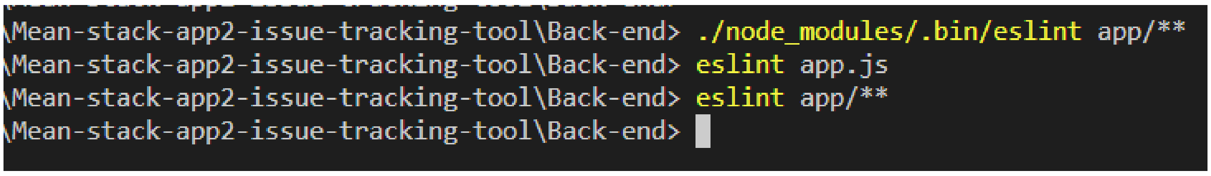
9. **Technical concepts used in back-end**
   - Followed MVC architecture
   - Middleware
   - Custom libraries
   - Socket.io
   - Nodemailer for sending mails
   - Apidoc for REST API documentation

10. **Code quality**
    - **Front-end** – All files pass linting.



```
PS E:\projects\mean-stack\Mean-stack-app2-issue-tracking-tool\Front-end> ng lint
Linting "issue-tracking-tool"...
All files pass linting.
PS E:\projects\mean-stack\Mean-stack-app2-issue-tracking-tool\Front-end>
```

- **Back-end** – All files pass linting.



```
\Mean-stack-app2-issue-tracking-tool\Back-end> ./node_modules/.bin/eslint app/**
\Mean-stack-app2-issue-tracking-tool\Back-end> eslint app.js
\Mean-stack-app2-issue-tracking-tool\Back-end> eslint app/**
\Mean-stack-app2-issue-tracking-tool\Back-end> 
```

11. Front-end application folder structure

- <u>App</u>
  - <u>Core module</u>
    - **Constants** – hard coded strings used in app
    - **Interfaces**
    - **Services**
      - **Auth:** signup, login, forgot/reset password API calls.
      - **Config:** get configurations from assets/config file.
      - **Export -** export issue reports in excel file.
      - **Error-interceptor:** Centralized all API call error handling.
      - **Token-interceptor:** Passing authToken to every API call request from one common service.
      - **Loader-interceptor:** Centralized show/hide loading spinner on waiting for all http calls
      - **Issue:** create, update, delete issue API calls
      - **Loader:** show/hide loading spinner
      - **Socket:** manage issue notifications via socket.io
  - <u>Feature</u>
    - **Authentication Module**
      - **Forgot-password component**
      - **Login component**
      - **Reset-password component**
      - **Signup component**
    - **Dashboard Module**
      - **Dashboard component**: manage dashboard page
    - **Description Module**
      - **Description component:** manage issue description page
  - <u>Shared module</u>
    - **Header component:** Shared across the app to display header
    - **Issue-form component:** Shared for both create and update issue
    - **Libraries module:** import and export third party libraries used in app
    - **Loader component:** Shared loading spinner across the app.
    - **Online-notification component:** Shared notification modal popup.

12. Back-end code folder structure

- <u>App</u>
    - **Controllers**
        1. **fileController.js:** Functions for upload, read, delete attachment files of issue.
        2. **issueController.js:** Functions for create, read, update, delete issue.
        3. **notificationController.js:** Functions for create and read notifications.
        4. **userController.js:** Functions for signup, login, forgot-password, reset-password.
    - **Libs**
        1. **checkLib.js:** Library to check null or empty value
        2. **emailLib.js:** Library to send email notifications
        3. **loggerLib**.js: Library to log errors and info while executing the application.
        4. **passwordLib.js:** Library to compare password
        5. **responseLib.js:** library to generate api response in a unique pattern
        6. **socketLib.js:** library for online meeting updates and notifications via socket.io
        7. **tokenLib.js:** Library to manage authorization token
        8. **validationLib.js:** Library to validate value based on conditions
    - **Middlewares**
        1. **auth.js:** Check the request whether it is authorized or not by verifying the authToken.
        2. **fileStorage.js:** Store the file in database
        3. **validation.js:** Validate the request parameters
    - **Models**
        1. **auth.js**: Database schema for Auth table
        2. **issue.js:** Database schema for Issue table
        3. **notification.js:** Database schema for notification table
        4. **user.js:** Database schema for User table
    - **Routes**
        1. **fileRoute.js**
        2. **issueRoute.js**
        3. **notificationRoute.js**
        4. **userRoute.js**
- **Config**
    - **AppConfig.js:** application configurations