Develop a full stack web-application Ticket management system (TMS) with following modules and features:

1. User login and Authentication

- There will be two types of Users - System Users and Organisation Users.
- System Users can login using email_id and 15-minutes valid 6-digit OTP sent on the email.
- System Users are admins for the entire TMS system and has access to all organisations.
- System Users can create, edit, view, delive 'organisations' inside TMS. An 'organisation' is any entity that will use TMS.
- 'Organisation' entity will have following fields:
    - organisation_name: unique identifier of the organisation
    - Name: organisation display name
- System Users can also create Organisation Users and add it to created organisations. Multiple users can be added to one organisation, but all these users in the same organisation should have unique email_id. However, different organisations can have same user with the same email_id.
    - Ex: An organisation user with email john.doe@gmail.com can be part of organisations with organisation_name org001, org002, org003. But org001 cannot have two users with same email john.doe@gmail.com
- An 'Organisation User' entity will have following fields:
    - Id: Unique id of the user
    - Email_id: Email Id of the user
    - First Name
    - Last Name
    - Date of Birth
    - Organisation Joining Date
- An organisation user can login using email_id, 15-minutes valid 6-digit OTP sent on the email, and the unique organisation_name that they are part of. If an organsation user tries to login to an organisation that they are not part of, error will be thrown.
- Organisation user cannot login into a delived organisation. If an organisation user is already loggedin, they cannot perform any activity and all the APIs should throw error.

2. Organisation Features:

- Any organisation user can create Tickets and assign it to other members of the same organisation. They can also assign following fields
- Tickets have following fields. Their requirement and form field details are mentioned together.
    - Type: [Story, Task, Bug] - Required. Will be dropdown.
    - Key: Unique identifier created using organisation_name and incremental number. ex: For organisation with organisation_name ORG001, ticket keys will be ORG001-1, ORG001-2, ORG001-3 etc Required. Auto generated.
    - Summary: Title of the ticket, task, content etc Required. Will be textarea.
    - Description: Description of the ticket, content. Will be textarea.
    - Assignee: Organisation User to whom the ticket is assigned Required. Will be dropdown consisting of organisation users of that organisation only.

    - Reporter: Organisation User who created the ticket Required. Will be dropdown consisting of organisation users of that organisation only.
    - Status: [TOBEPICKED, INPROGRESS, INTESTING, COMPLETED] Required. TOBEPICKED will be assigned when ticket is created.
    - Created date: Required field. Auto added.
    - Updated date: Required field. Updated whenever there is any change.
    - Due Date: Deadline of the ticket. Optional field
    - Files: Any file of form png, jpg, jpeg, pdf, mp4 can be attached to the ticket. File Uploader. Optional field
- Tickets from one organisation should not be visible to another organisation.
- Post creation of Tickets, apart from 'Key', 'Created date', 'Updated date' everything else can be changed by any organisation user.
- An organisation user will have a page to view the list of Tickets assigned to them.
- An organisation user can view list of all tickets and apply filters on them. Filters can be applied on 'Status' (dropdown with all possible values), 'Type' (dropdown with all possible values), 'Status' (dropdown with all possible values), 'Created date', 'Updated date', 'Due Date'. 'Assignee', 'Reporter'. By default, filter will be applied on 'Assignee' with them as the filter i.e by default tickets assigned to them will be visible.
- Ticket 'Summary' can be maximum 250 characters long, but on the list page only first 10 words should be visible. Clicking on 'Summary' should take the user to Ticket details page.
- On Ticket Details page, all details of Ticket will be visible. Except for 'Key' and 'Created date', all fields are editable by organisation users on this page. Enhance your UI
- Under ticket details main section, there will be comment section, where organisation users can add comments by adding text in a textbox and pressing 'Enter' to add comment. The comment text along with user details and created date will be recorded against the ticket. All organisation user can then see this comment on the ticket. The commenter user can also edit, and delete the comment.
- Any organisation user can also sort the comments based on 'Newest first' or 'Oldest first' criteria.
- Ticket History: Any change made to the Ticket by organisation user will be recorded in its history in form of "{User Name} updated the field {Field Name} – from {Old Value} to {New Value}
  ex- John Doe updated Status – from TOBEPICKED to INPROGRESS

3. Technology Stack:

- NodeJS [v18 or above]
- JWT for auth
- ExpressJS
- TypeScript
- MongoDB [v4.4 or above] with Mongoose
- Redis for caching frequently used data points in your application
- Zod for validations
- React
- Redux
- SASS
- HTML5, CSS3, JavaScript
- RSuiteJS as CSS library

4. Backend Folder Structure - Please follow the strict folder structure as mentioned below:

- 'src' folder should contain all your codebase
- package.json to contain all the packages used
- Inside 'src' folder, following folders should be present
    - controllers – all the api controllers will reside here

    - services – controller function will call function in services folder and all the logic will be written in services folder
    - constants – all constants will be declared here
    - dao – all the logics for accessing db will go here
    - models – model schema for db access will be declared here
    - routes – all the api routes will be declared here
    - app.ts – entry point to your application
    - formatters – function to format request response will be declared here
    - typings – global types for entities will be declared here
- Reference Links:
    - Typescript: https://www.typescriptlang.org/docs/handbook/intro.html
    - Express JS: https://expressjs.com/en/guide/routing.html
    - Redis: https://redis.io/docs/latest/get-started/

5. Frontend Folder Structure – Please follow the strict folder structure as mentioned below:

- 'src' folder should contain all your codebase
- Package.json to contain all the packages used
- Inside 'src' folder, following folders should be present
    - atoms – all the smallest unit components such as Button, Input etc will go here
        - Each component should have three files – tsx file for component, css for styles, types.d.ts file for props type of that component
          ex: Button > [Button.tsx, Button.module.scss, types.d.ts]
    - molecules – all the UI elements/components created from atoms will go here. No API call should happen here, and its only meant for creating UI
    - organisms – components comprising of atoms and molecules should go here. Here APIs can be called and workflows created
    - pages – different pages of your application will go here, strictly pages only
    - services – all the API declarations should go here
    - typings – global types of your application will go here
    - routes.ts – all the route urls for your application on which different pages will be rendered goes here
    - app.ts – starting point of your application
- Reference Links:
    - Typescript: https://www.typescriptlang.org/docs/handbook/intro.html
    - RsuiteJS: https://rsuitejs.com/components/overview/
    - React Router DOM for routing: https://reactrouter.com/en/main
    - SASS: https://sass-lang.com/documentation/