Draft Scope
for
Ticketing System
assigned to
Angelique Taute

Table of Contents

Quick Overview

This document will outline the project specifics for the design, technological aspects, and application requirements for the ticketing system. The ticketing system should consist of submitting, organizing and viewing tickets.

JavaScript frameworks will be required as the main programming language with an open-source database to evaluate your skills with database design and experience with MVC's. Users will need to register an account to be able to view and post tickets based on clients, projects and systems, which will be discussed in more detail.

The goal is to create an application that works seamless, is easy to use and can act as a work in progress and progress report for users.

If you require any assistance with technology compatibilities or have any questions related to requirements, feel free to contact barendo@sabttec.com for more clarity.

Project Description

We would like to create an application for creating and viewing tickets based on levels consisting of clients, projects, and systems. These tickets should be easy to navigate for historical record keeping purposes, which should contain information as to who worked on these tasks, when it was completed and for whom it was assigned to (client, project and system). Having multiple ways to filter the tickets will also prove as an effective report functionality.

- ### Programming language and MVC

The use of JavaScript Frameworks is encouraged to save time on development via the use of libraries, which contain predefined code for a variety of useful functions. By using JavaScript Frameworks, we get access to a wide variety of JS libraries which contain pre-written code for useful routine procedures, tasks or features. These code components can then all be added as individual code blocks to build your website, while remaining as separate entity. This also allows for better troubleshooting and improves productivity.

- ### Database

Databases such as MySQL and Postgres are recommended for storing data because they are both well-known RDBMS's which have compatibility with most modern-day languages and are open source.

- ### Functionality and features

Users should create an account with basic details such as name, surname, email, company, and contact number. Users should then be able to log in with their account details and then view tickets related to them on their home page.

These tickets should show a unique ID (reference code, more detail in requirements section), description of the task/problem, date submitted, status and type. The user should also be able to filter these tickets for more specific results which will be further explained in the requirements section.

Users need to be assigned to specific roles and permissions, which will allow admin users (predefined in database) to manage registered users by creating and assigning clients, projects, and systems to them. These permissions need to be linked as the following roles: admin, basic user and manager.

Creating a ticket should consist of a form with details such as client, project, system, type of ticket, short description and screenshot if applicable.

All created tickets need to have a status attached (Created , assigned , in progress , completed , rejected)

The cycle should follow: 1. Ticket created (Created)

2. Assign ticket to user (Assigned)

3. User accepts ticket and starts work (In progress)

4. Ticket marked as complete after confirmation of resolution.

5. If ticket is not relevant or incorrect status is set to rejected (Rejected)

When marking ticket as Complete or Rejected a mandatory reason should be captured.

Reports should be created to allow managers to pull reports via different criteria's which will be further explained in the requirements section.

Finally, there should be a requests section where users can submit requests to access certain sites or require authentication.

Requirements

- ### Pages/views

The following pages are given as a guideline for the core views we are looking for and can be expanded but should require the pages below. Names of these pages can be changed to your liking as long as it makes sense and are easily discernible.

1. **Login** *(landing page when opening the app)*
    - This page should be the first page the user sees when browsing to the app.
    - The page should have inputs for username/email and password.
    - Login button
    - Give feedback when login failed.
    - Link to registration page when required to create an account first.
2. **Registration**
    - The user creation page, where users create their own account with basic information such as name, surname, company, email, password, confirm password and contact number.
    - Email authentication is not required.
    - User feedback for successful account creation and a link to the login page.
    - Also give feedback (error message) if the account was not created.
3. **Header**
    - This page will be required in all your pages and will act as the top section of your application.

- This is where you would usually put the logo and name of your app (which you can make up or use existing names as temporary placeholders) and your profile menu to edit your details or logout.
- When clicking on edit profile, you should be redirected to the my-profile page.
- When clicking on logout, your session should be destroyed and redirect you to the login page.
- The look, styling and positioning are left to your own creativity, a basic display is more than fine.

4. **My-profile**
   - This page is used to edit your own details, but only the details you entered during user registration.
   - Clients and projects are only allowed to be added from manage users and should be an admin or role that have permission to do so.
   - You can decide if you want to add a profile picture.
   - When changing your password, let the user type their old password, enter a new password and re-type their new password as a counter measure for unauthorised access.
5. **Side-menu**
   - This menu, as with the header should always show on the application, preferably on the left side.
   - Menu items should be clickable and navigate to their respective pages.
   - This will act as your main navigation through the application.
   - Menu items consist of the following:
   - Tickets (default display when logging in but can be clicked to act as the go “home”)
   - Create/submit ticket
   - Clients and projects
   - Manage users
   - Reports
   - Create request
   - View requests
6. **Home/index**
   - This will be your landing page after you successfully logged on.
   - Show tickets assigned to clients, projects and systems you are part of.
   - Table view would be the preferred method to show a list of tickets but should be limited to ten at a time.
   - Pagination should be used to go to the next page and show the next ten results and so on.
   - By using tables, you also get access to column filters of the data you display, this filters usually only filter alphabetically, which means that you should add a custom filter for dates.
   - Depending on which tables you use, plugins or standard html built-in tables, you could also have a search bar to filter results in your table.
   - Each ticket should have a status which shows the progress of the ticket. These statuses should consist of pending, in progress, testing and resolved.
   - The logged in user should be allowed to set the status of each of these tickets by selecting from a dropdown select with the various statuses.
7. **Create-ticket**

- A form with all the details for your ticket.
- Feel free to add additional info but the form should contain the following:
- Client, project, system, type of ticket (bug, feature, hotfix, trap, db change), description and image if applicable.
- When submitting the ticket, use the logged in user to store the email of the user as the ticket creator and the current timestamp to show when the ticket was created.
- Also generate a unique code which we will call the reference code/id and should consist of the following parts: [client]-[project]-[system]-[pk] ("STec-DevH-IntCal-23") – first part will be the abbreviation of the client, second part will be the abbreviation for the project, followed by the abbreviation for the system and finally the incremented number, which could be the primary key of the ticket.
- This reference key will be used when recalling a specific change because most divisions will use this code in their comments.
- All tickets will have a status attached (Created , Assigned , In progress , Completed , Rejected)

8. **Clients, projects and systems**
   - Only admin users and managers should have access to this page.
   - This is where you will create clients, projects and systems.
   - You should also be able to view them all and how they are linked.
   - Allow edits and deletes.
   - All entries here will be the available options when creating a ticket for their relevant selects which consist of clients, projects and systems.
9. **Manage users**
   - Admin and managers should be able to assign users to clients, projects and systems.
   - Users are allowed to be link to more than one client, projects and systems.
   - Just make sure that when assigning clients to a user that only projects belonging to that client gets retrieved, the same for when selecting a project.
   - Assignment of these entities would probably be best done in steps. First selecting client, then show a list of projects belonging to that client. Once a project is selected, show all systems belonging to that project. Projects and systems could be multiples per client.
   - Once finished with all the assignments a button should be clicked which will save all selected items and link them to the user.
   - Additional clients, projects and systems should be allowed for users, which means that users can belong to multiple clients, projects and systems.
   - The main goal of restricting access to this page is to only allow specific clients, projects and systems for a user.
   - You can also use this page to set the roles of each user.
10. **Reports**
    - All users should be able to generate reports based on a criterion.
    - Limit access for basic users and managers to only generate reports based on their linked clients, projects and systems.
    - Admins should have no restrictions.
    - When the filter is filled in a button should be pressed to generate a report, which should show the results in a table view below the filter. Again, limit the results to 10 and add pagination for the rest if the results.

- There should be a button to export/download pdf that consist of all the results based on the generated report.

11. **Create-requests**
    - Requests should consist of a form where the user can submit a question, issue they have or require authentication from someone to grant them access to a system, voice channel, chat room etc.
    - Here the user should choose a request type from a list of keywords, such as: access, authentication required, validation, more info, testing required etc.
    - The user should add a short description of what they require or are requesting.
    - Clients, projects and systems should also be options if the request is related but does not have to be required.
    - Additional info should be an input field where a user may type a system name if it is not included in the list of systems and can also put application names such as discord or gmail if access is required. In case of an existing ticket the ticket reference can be added here.
    - The signed in user should be stored as the sender/requester.
    - Time stamp of the current time should be used as the date created.
    - This page should act similar as a contact us page where users can ask for more info.
12. **View-requests**
    - Only admins and managers should be able to see these requests.
    - All details should show in a table for admins and managers to take actions from here.
    - Managers and admins should have a button to click on, which changes the status of the request from pending to resolved.
    - For now, we can say that they need to manually email the requester instead of sending an automated email to them about their request being resolved (this can be added as an additional feature but is not required since this is only a demo version). Only status update is required for the demo version.

DB design

Most of the following names, tables and data types are only given as examples and can be expanded as you see fit or require additional data. You can call the database anything you like if it is relevant to the application and understandable. Try to keep it simple and to not include too many special characters and camel casing. Stick to a specific naming convention and try to avoid spaces. When using 2 words, try to use dashes ('-') or underscores ('_') to combine the word. Also decide to use either all caps or all lower case.

1. **Database name**
2. **Database tables**:

   Users
   - id
   - Name
   - Surname
   - Email
   - Password
   - Contact nr

- Company
- Role (default as basic user)

Tickets

- Id
- Type
- Description
- Client
- Project
- System
- Status
- Reference
- Creator (email of the person who created the ticket)
- Image
- Date created

Ticket_types

- Id
- Name (bug, feature, hotfix, trap etc.)

Clients

- Id
- Name
- Abbreviation

Status_types

- Id
- Name (Created, Assigned, In progress, Complete, Rejected)

Projects

- Id
- Name
- Abbreviation

System

- Id
- Name
- Abbreviation

User_tickets *(associative table to link users to specific clients, projects and systems)*

- Id
- User
- Ticket (from the table: client_project_systems)

Roles

- id
- name

Permissions

- id
- function
- description (can be null)

Requests

- id
- type

- description (can be null)
- client
- project
- system
- additional info
- requestor
- requested date
- status (default 0, changes to 1 when resolved)

Role_permissions *(associative table to link permissions to a role)*

- id
- role
- permission

Client_project_systems *(associative table to link systems to projects and projects to a client)*

- id
- client
- project
- system

- ## Features and Functionality

You could use the following list as a check list to see if you included all the required functions and features for the application.

1. First page: login
2. User creation
3. User input validation for password and email.
4. Password hashing
5. Show error messages and feedback when account have been created.
6. Session creation
7. Login with created user.
8. Manually creating roles in the database: admin, manager and user.
9. Manually creating permissions in the database.
10. Manually linking the roles with each permission in the database.
11. Creating a default admin user which you could use for testing.
12. Make sure to have 3 different users assigned to 3 different roles which you could use for testing permissions.
13. After logged in show a list of tickets assigned to the user, display appropriate message if no tickets exist.
14. Only allow account creation details to be altered when using edit profile.
15. Show feedback when details have been updated or error message when failed.
16. Show only tickets the user is assigned to.
17. Test filtering of tickets and make custom filters where needed.
18. Add a button or select to change the status of a ticket.
19. Check if logout works, sign out and try to manually browse to the home page where you logged out. When manually redirecting to a page, a validation check should be used to automatically redirect the user back to the login screen, but should work if the user is logged in.
20. Make sure that all links in the side menu and edit profile redirects to the correct pages.

21. When navigating from the menu, make sure that the header and side menu carries over to the new pages.
22. Test the create ticket.
23. Generate reference code as stated in requirements->pages->7 section to consist of 4 sections namely: [client]-[project]-[system]-[pk].
24. Use timestamps as date created for ticket and save current logged in user as creator.
25. Effectively create clients, projects and systems.
26. Able to link systems to projects and projects to clients.
27. Make 3 different clients, which all have at least 1 project linked to them, one with multiple projects as well as contain a minimum of 1 system each and one with multiple systems.
28. Set the manager role from the manage users page and test if it works.
29. Assign clients, projects and systems to the users. Test multiple as well.
30. Create reports with and without filters (without filters should show all tickets).
31. Test all filtering in the reports.
32. Export to pdf and make sure it works and is structured.
33. Test if permissions apply to admin, manager and basic user when generating a report as stated in requirements->pages->10.
34. Create request
35. Store requestor/creator as the logged in user and timestamp as date requested.
36. Test to see if admins and managers can view the created requests.
37. Allow admins and managers to set status of requests to resolved.

• Additional Features

The following features are not required for the demo and can act as additional features to make things easier.

- Email ticket creator and requestor when their ticket or request have been resolved.
- Allow managers and admin to assign tickets to other users / managers. Duplicate or move from one to another.
- Allow users to add comments on their tickets to give more insight of what they did if they want to.
- Add priorities on tickets, which users could set and arrange accordingly.
- Add deadlines on tickets.
- Include and capture timestamps on each status update.
- Allow admin to create teams consisting of managers and users, which will allow further refining of specific tickets based on teams.

• Styling

We are not concerned with a very stylized application and color schemes, pictures, icons etc. The application should comply with usability standards which is user friendly, structured, readable, and not overly complicated. Styling would be encouraged to help structure the readability of the application and allow the app to “flow” efficiently when doing tasks or working with functions. When using colors try to steer clear from too bright text such as red, green and too dim in contrast with the background color such as light grey and yellow. The use of pictures, icons, logo’s, color schemes/patterns and naming conventions are completely up to you, as long as it fits the usability of the application. Styling will matter with the landing page(login) and the registration page. It should look like a typical login and registration page, structured correctly and aligned appropriately.

- ### Plugins and third-party applications

The use of plugins and other third-party applications are allowed but try to stick to community (free to use) options. The use of bootstrap is also encouraged for structuring and styling where applicable.

Closing remarks

This application does not have to be perfect because this is a demo version, and it can also exclude some features if the main functionality is there. Remember that all examples listed in the explanations are subject to change and only act as a guideline. Feel free to make changes where you see fit. If you feel uncertain with any task or require more clarity in regards of a function, please feel free to contact barendo@sabttec.com for more information.