BUG TRACKING SYSTEM

& Documentation

ABSTRACT

This specification describes how to create an application that tracks the assignment, status, and progress of issues related to a project. The scope has been deliberately limited due to time constraints. I would however encourage you to add features and add this project to your repositories on GitHub.

R. Hood

Web Programming 2x1

TABLE OF CONTENTS

BEFORE YOU BEGIN

1. There will only be a single sign-on for the system, and this will be an admin account.
2. Use Web Storage API (localStorage) to persist data. WebAPI is a mechanism for storing values in a browser-based database. You will find links on how to use this API at the in the Resources section at the end of this document. You are not required to enforce any relationships between the objects in your localStorage; this is a demo application.
3. There are no specific requirements for the user interface. You must build the UI in an intuitive way.
4. You may use any UI library e.g., Bootstrap to create the UI, but you may not plagiarise a solution.
5. There is a markdown component to this project. I have included useful links in the Resources section.

INTRODUCTION

WHAT IS AN ISSUE TRACKING SYSTEM

> *A bug tracking system or defect tracking system is a software application that keeps track of*
>
> *reported software bugs in software development projects. It may be regarded as a type of issue tracking system.*
>
> *Many bug tracking systems, such as those used by most open-source software projects, allow*
>
> *end-users to enter bug reports directly. Other systems are used only internally in a company or organization doing software development. Typically bug tracking systems are integrated with other project management software.*

> *A bug tracking system is usually a necessary component of a professional software development infrastructure, and consistent use of a bug or issue tracking system is considered one of the "hallmarks of a good software team".*

> [https://en.wikipedia.org/wiki/Bug_tracking_system]

All issues will be created as tickets in your application. The next section provides some definitions.

DEFINITIONS

Bug

A software bug (or just “bug”) is an error, flaw, mistake, failure, fault, or “undocumented feature” in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code.

Ticket

A ticket is where you would report a bug that you have discovered during the project.

Markdown

Markdown is a lightweight markup language for creating formatted text using a plain-text editor. You will create a markdown document as an additional page that will be accessible as part of your project deliverable. Use the markdown to create documentation for your system.

REQUIREMENTS

AESTHETIC REQUIREMENTS: SYSTEM INTERFACE

The system must be used via a web interface created using markup. Here is an example of an interface that can be used to build the system. It is just a sample, and you must only use it to visualise the kind of system that is required. There are plenty of other examples that you can look at for ideas. The picture displayed below only shows the summary of issues.

Once an issue is clicked, details of the issue must be displayed in an intuitive fashion. More on this later in the functional requirements.

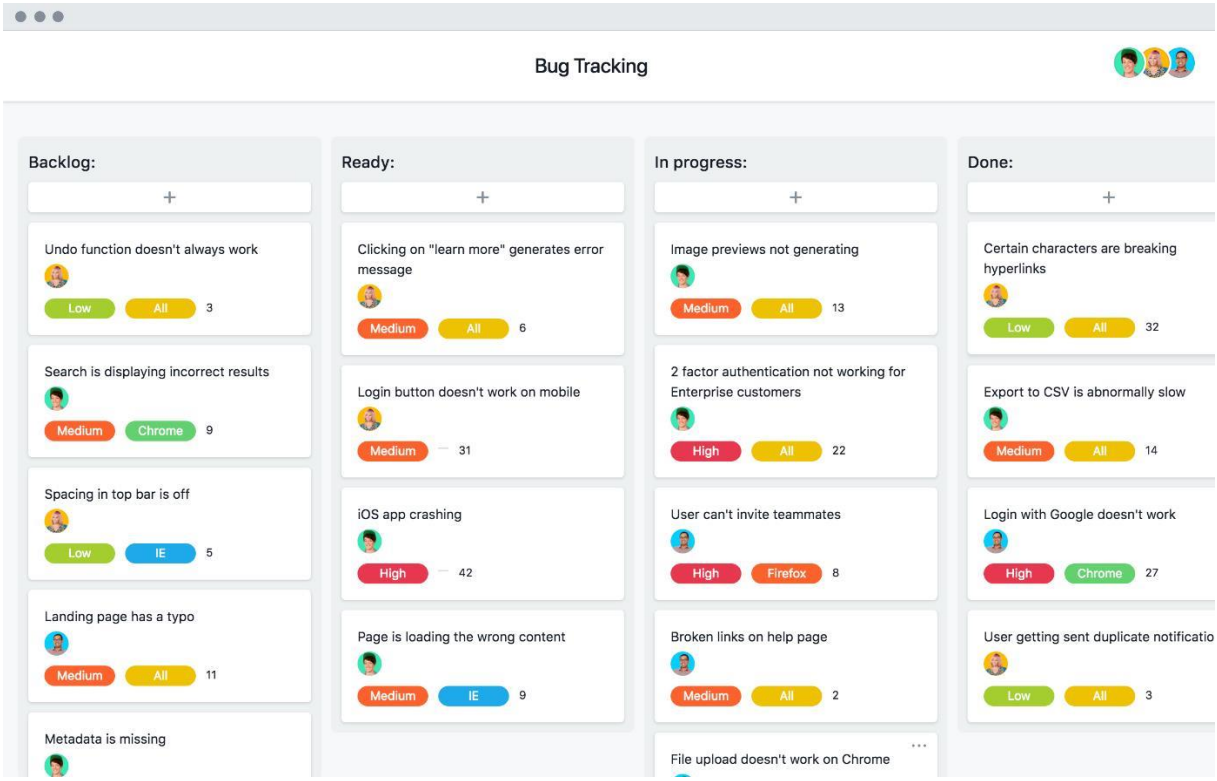

Figure 1: Sample UI for a bug reporting system showing the summary of bugs and the status (as headings)

FUNCTIOANAL REQUIREMENTS

HIGH LEVEL REQUIREMENTS

The system must allow the user to:

- Create issues
- Assign issues
- View all issues
- View a particular issue
- Edit issues
- Create people [1]
- Create projects[2]

CREATING AN ISSUE (TICKET)

The following information is required:

- Summary of the issue that was discovered
- Detailed description of the issue
- Who identified the issue
- The date on which the issue was identified
- Which project the issue is related to

- Who the issue is assigned to
- A status of the issue [open/ resolved/ overdue]

[1] You can use static values in the database as opposed to using a form.

[2] You can use static values in the database as opposed to using a form

- Priority of the issue [low/ medium/ high]
- Target resolution date
- Actual resolution date
- Resolution summary

ASSIGNING AN ISSUE (TICKET)

- Once all the ticket information has been recorded, you must *select* a person to fix the issue.
- It is possible to create a ticket and assign the issue to a person later.

VIEW ALL ISSUES (TICKETS)

- See the example in figure 1, showing limited details for all issues.
- Consider how the UI could be affected if there are too many issues that have been recorded.
- Decide on the details that make sense

VIEW AN ISSUE (TICKET)

- The system must display all the details for that issue. See Creating an issue.

EDITING OF AN ISSUE

- Summary of all issues
- Detailed issue
- All issues by project **Bonus*

CREATING PEOPLE

- A person must have an id, name, surname, email address, and unique username. Add a profile picture for bonus marks.

CREATING PROJECTS

- A project must have an id and name. A name is any valid string.

DATA REQUIREMENTS

The following data needs to be persisted across sessions:

- **Projects:** A list all current projects
- **People:** A collection of information about who can be assigned to handle issues
- **Bugs**: Information about an issue, including the project to which it is related, and the person assigned to the issue.

SUBMISSION

- Submit the final solution on MS Teams. Only one person should submit on behalf of the group. This information must be completed on the Excel rubric once teams have been finalised.
- Add this project to your GitHub repository. This is not for marks, but to build a digital portfolio that you can showcase to potential employers in the future. [3]
- Marks will be awarded based on your in-class presentation. I will share a rubric at a later time, but you should use the requirements to guide you.
- Ensure that you populate your app with at least 20 issues for testing purposes.Vary the following attributes:
    - Dates [entry date, resolution date, and target date]

[3] You would need to add features because this scope is exceedingly limited.

o Priority level o The person responsible for resolving the issue.

RESOURCES

WEBAPI

https://javascript.info/localstorage https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage https://www.section.io/engineering-education/how-to-use-localstorage-with-javascript/ (contains comprehensive examples)
https://blog.logrocket.com/localstorage-javascript-complete-guide/
https://www.makeuseof.com/localstorage-javascript/

MARKDOWN

https://www.markdownguide.org/getting-started/ https://medium.com/analytics-vidhya/how-to-create-a-readme-md-file-8fb2e8ce24e3 https://document360.com/blog/introductory-guide-to-markdown-for-documentation-writers/ https://www.markdownguide.org/cheat-sheet/
https://www.freecodecamp.org/news/markdown-cheat-sheet/ (*how to create and preview markdown in VS Code*)

BOOTSTRAP

https://getbootstrap.com/docs/5.2/getting-started/introduction/
https://getbootstrap.com/docs/4.3/components/ (Contains awesome resources to build your UI **very quickly**)
https://bootstrapious.com/p/admin-template (example of a template you can use to customise your interface)

EXAMPLES OF BUG TRACKERS

https://www.youtube.com/watch?v=PYxLwn-Kk2U
https://www.youtube.com/watch?v=vG824vBdYY8
https://www.youtube.com/watch?v=VT6k-lJO7WM
https://usersnap.com/blog/open-source-bug-tracking/