**Scope of the project:**

A simple web-based Bug Tracker System designed to manage and track the lifecycle of bugs for multiple projects. Named Syn (Norse deity of watchfulness and refusal) Can be an acronym for System Yips Navigator?

It features user registration and authentication, project management, bug reporting with detailed descriptors, and the ability to update bug status and comment on issues.

1. **User Registration & Authentication:** Users must register and create an account to report bugs or manage projects. (Admin users should assign roles and approve registrations?)
2. **Project Creation & Management:** Approved users can create new projects for which bugs will be tracked. They can also view and manage their existing projects. (Project manager can create projects only?)
3. **Bug Reporting:** Users can report bugs they encounter in a project. The bug report should include a title, description, severity level. (upload video/picture? Easier to link to 3rd party media hosting websites)
4. **Bug Tracking:** All reported bugs are listed in a project's dashboard. Bugs can be filtered and sorted by different parameters like severity, status, and date reported.
5. **Bug Status Updates:** Users can update the status of bugs (new, in progress, resolved).
6. **Discussion and Comments:** Each bug report should allow for discussion or comments, allowing users to discuss the bug, propose solutions, or give updates on resolution progress.
7. **Roles and Permissions**: Each user is assigned a different role with different permissions to modify the bugs and projects and status.

**Entities:**

* User: Represents the users of the system. Attributes can include userId, username, email, password, etc.
* Role: Represents the different roles that a user can have, such as "admin", "developer", "tester", "project manager", etc. Attributes can include roleId, roleName.
* Project: Represents the software projects in your system. Attributes can include projectId, projectName, description, startDate, endDate, etc.
* Bugs: Represents the bugs reported in the system. Attributes can include bugId, bugTitle, bugDescription, status, priority, creationDate, lastUpdated, etc.
* Comment: Represents comments made on a bug by users. Attributes can include commentId, content, commentedOn, userId, bugId.
* BugHistory: Represents changes made to a bug. This can include changes to the status or priority of a bug, assignment changes, etc. Attributes can include historyId, bugId, changedOn, oldValue, newValue, etc.
* BugAssignment: Represents the assignment of a bug to a user. Attributes can include assignmentId, bugId, userId, assignedOn.
* ~~Module: This can represent different modules or components of a project. Each bug can be linked to a particular module. Attributes could be moduleId, moduleName, projectId, etc.~~
* ~~Tag: Represents different tags that can be associated with a bug. For example, "UI", "backend", "security", etc. Attributes can include tagId, tagName.~~
* ~~BugTag: Represents the relationship between Bug and Tag if you allow a bug to have multiple tags. Attributes can include bugId, tagId.~~
* ~~Attachment: Represents any files attached to the bug report, such as screenshots, error logs, etc. Attributes can include attachmentId, fileName, fileType, bugId.~~
* ~~Notification: Represents notifications sent to users about changes to bugs they are interested in. Attributes can include notificationId, userId, message, sentOn.~~
* ~~ProjectUser: If you need to assign users to specific projects, this class represents the relationship between Project and User. Attributes can include projectId, userId.~~
* Permission: Represents different permissions a user might have, depending on their role. Attributes can include permissionId, permissionName.
* RolePermission: Represents the relationship between Role and Permission. Each role can have multiple permissions. Attributes can include roleId, permissionId.

**UI Flow:**

1. **Landing Page:**
   1. A simple, clean design with the name of the bug tracking system at the top (Syn).
   2. Two main buttons: "Register" and "Login."
   3. If users are logged in, these options would be replaced with "Dashboard" and "Logout."
2. **Login Page:**
   1. Fields for the username and password.
   2. A "Login" button to submit the form.
   3. A "Back" button to return to the landing page.
   4. A "Forgot Password" link.
3. **User Dashboard:**
   1. The user's name at the top, with an option to view their profile.
   2. A list of the user's reported bugs, with columns for title, severity, status, and date reported.
   3. A "Report Bug" button that leads to the Bug Reporting Form.
   4. Each bug title in the list is a link to the Bug Details Page.
   5. If user is “Admin”, a button to manage other users with a popup
4. **Bug Reporting Form:**
   1. Fields for bug title, description, and a dropdown to select severity level.
   2. A "Submit" button to submit the form.
   3. A "Back" button to return to the dashboard.
5. **Bug Details Page:**
   1. Displays the title, description, severity, status, and date reported of the bug.
   2. If the user viewing the page is the one who reported the bug, they will see an "Update Status" button.
   3. “Admin” and “Developer” have buttons to resolve and close the bug.
   4. A "Back" button to return to the dashboard.
6. **User Profile Page:**
   1. Displays the user's name and email, and a list of bugs they've reported.
   2. Each bug title in the list is a link to the Bug Details Page.
   3. A "Back" button to return to the dashboard.
7. **Forgot Password Page:**
   1. A field for the user to enter their email.
   2. A "Submit" button to submit the form.
   3. A "Back" button to return to the Login Page.

**Database Design:**

