Application for shortening URLs and for using those short links is pretty straightforward. That kind of apps is fairly common over the Internet and from the user's perspective, they all work the same. The use case is simple:

1. User pastes the link in the input field of the application
2. User confirms procedure
3. Shorten link is ready and visible for the user to read and/or copy

All links from one provider have the same short and easy to remember form:

`some.domain/short-id`

Often when a link shortening app belongs to some other company or it's related to some website, the domain part is also shortened by reducing its name by few letters and putting a dot for separating the app domain from TLD (top-level domain). Although the domain name is not a matter of this paper.

Part of the link we're interested in is id. The main issue with this is to generate an id that will be unique for every shorten URL in our application, but it will also be compact and eventually possible to remember. With that in perspective, a good choice will be a short combination of letters in different cases and numbers. This set of characters is commonly known as Base62 encoding. Depending on the usage of this shortener, the length of the id can be easily adjusted. With a length of 5 characters, we have more than 900 million unique ids. With that length and generating 5000 links per day, it would be last for the next 500 years. So no matter the usage, 5 characters of length should is fine.

UI of this application should be simple and responsive. Taking into consideration using this app as a widget of another webpage, it should be good-looking and usable in small size. All elements of UI should be as follows:

1. Header with a name of the application (for accessibility reasons)
2. Text field for URL input. It should also be used for returning shorten links, so space on the webpage can be used efficiently and users can easily copy this link to clipboard.
3. This text field should be equipped with a label that would help users use this app properly.
4. A button that would be the only control element used for requesting a link shortening service and for resetting the process.

Considering user experience, UI should handle edge cases like invalid user inputs e.g. given input is not a proper URL or cases where the backend side of service is not available at the moment. Those situations should be explicitly presented to the users. On the invalid input text field, the border color should be changed to signal this, and submit button should also be disabled. Issues on the service side should be displayed on the text field label.

Backend should consist of two controllers:

1. Controller for POST method with query parameter consisting URL to be shortened
2. Controler for GET method with the id parameter that would redirect the client to proper URL for given shortened link

For a POST request backend should read the database and check if the requested URL is already present on the system. If not, a new unique id should be generated and saved in the database paired with a URL. The request ends with sending an id matching the given URL.