Project name	CloudCooking

1. Short project description (Business needs and system features)

This is a site for users to share their cooking recipes with others and comment on which ones they like or not. The system will be developed using ***Golang***. Front-end will be made with ReactJs. Axios will be used for async requests, MobX for state management, React router for routing, Semantic UI for components. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization. Server side routing with ***go-chi***. Database used will be postgres with Gorm (***https://gorm.io***). Global chat with github.com/gorilla/websocket. The main user roles (actors in UML) are:

- *Anonymous User* – can only view homePage, login and register
- HomeCook (extends *Registered User)* – can add their own recipe and comment on other people's recipes

2. Main Use Cases / Scenarios		
Use case name	**Brief Descriptions**	**Actors Involved**
2.1. **Browse recipes and view site information**	All Registered Users can view recipes and information views	*Registered Users*
2.2. Register	*Anonymous User* can register in the system by providing a valid e-mail address, username and password. By default, all new registered users have *HomeCook* role.	*Anonymous User*
2.3. Login	*Anonymous User* can login in the system by providing a valid e-mail address and password	*Anonymous User*
2.4. **Chat**	Registered users can chat in a public chat.	*Registered Users*
2.5. Add/Edit Recipe	Registered users can create their own recipe and edit it.	Registered Users
2.6. Delete Recipe	Registered users can delete their own recipe	Registered Users
2.7. Comment on recipe	Registered users can comment on other users recipes	*Registered Users*
2.8. View recipes list	Recipes are sorted by time added	Registered Users

3. Main Views (Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. **Home**	Shows basic information of the site and allows users to login and register	/

3.2. **View recipes list**	A page for viewing recipies	*/recipes*
3.3. View detailed recipe information with comments	This view shows all of the information about the recipe and shows the user comments. The user can also add his own comment	*/recipes/{recipeID}*
3.4. **Manage Recipe**	This view allows the user to manage his created recipe	/manage/{recipeID}
3.5. Create Recipe	A view for creating new recipe	*/createRecipe*
3.6. **Chat**	Allows users to chat with each other	/chat

4. Database Entities / API Resources		
View name	**Brief Descriptions**	**URI**
4.1. Users	GET *User Data* for all users, and POST new *User Data* (Id is auto-filled and modified entity is returned as result from POST request).	*/api/users*
4.2. User	GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions described in UCs.	/api/users/{userid}
4.3. Login	POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests.	*/api/login*
4.4. **Register**	POST *User Credentials* (e-mail address, username, password) and receive a valid *Security Token* to use in	*/api/register*

	subsequent API requests.	
4.5. Recipies	GET, POST Recipes	*/api/recipes*
4.6. Recipe	GET, POST, PUT, DELETE recipe by recipeId	*/api/recipes/{recipeId}*
4.7. **Recipe comments**	GET comments for a Recipe with recipeId	*/api/recipes/{recipeId}/comments*
4.8. **Comment**	PUT, DELETE Comment for *Recipe* with specified *recipeId* and Comment with specified commentId.	*/api/recipes/{recipeId}/comments/ {commentid}*
4.9. Categories	GET, POST categories	*/api/categories*
4.10. Category	*GET, PUT, DELETE Category with categoryId*	*/api/categories/{categoryId}*
4.11. **Chat**	GET, POST messages between all users who are connected to the chat	/api/chat