Team Project

Deliverable 2 – Requirements Specifications

CSCE 5430 (Fall 2021)

**Due Date:**

* Report (project repository**):** 9/30, 11:59pm (11/12-point, single space, 10-15 pages)

# Weight: 10% of the final grade

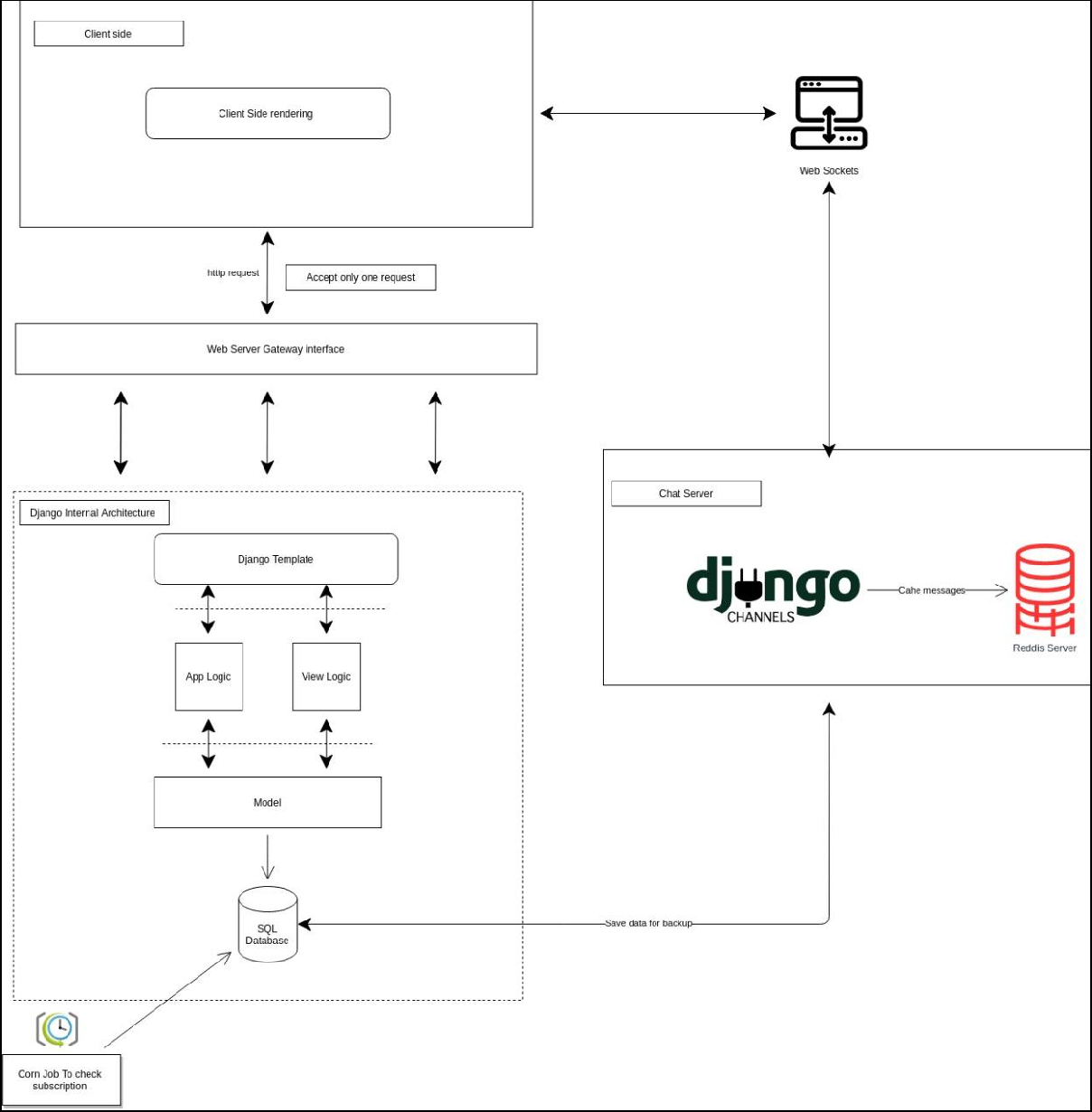
**Late Policy:** 10% deduction per day.

For the second part of the project, you will develop **a set of requirements specifications,** and make a plan for implementing the project through three development phases (an incremental approach). Your requirements specifications should describe all intended functionalities including non-functional attributes.

**Hand-in:**

1. The overall structure of the system. This should contain a diagram and descriptions about the system structure including individual components (subsystems).

**Structure:**



**Description:** Our project employs a server-side rendering (SSR) model, and instead of rendering it in the browser, we use JavaScript and the rest protocol on the client-side. When a client sends an HTTP request, WSGI accepts it one at a time and sends it to the Django framework, which uses an MVT architecture in which the view accepts the request and search response in the database (i.e., SQLite) and changes variables in the template with values. We utilize the Redis server to store messages/chat data because users have the option to interact with Designers.

**Components:**

**Web sockets:** When a user is authorized, Web Sockets (through Django Channels) manages the communication between the client and the server, and an event is published to all other connected users. The screen of each user will change without them having to reload their browsers.

**Client-side rendering:** With JavaScript and client-side rendering, we can render our websites entirely in the browser. A client-side rendering website builds each route dynamically in the browser, rather than having a separate HTML page for each route.

**Web Server Gateway Interface (WSGI):** A simple calling convention allows web servers to pass requests to Python-based web applications and frameworks. From there, the replies are subsequently forwarded to the webserver, which response to the requestor.

**Django Template:** Django's MVT Structure is completed by templates, the third and most significant portion. The Django framework efficiently handles and generates dynamically generated HTML web pages for end-users. We use templates to provide a frontend and layout to our website because Django is mainly a backend framework.

**Model:** The model will serve as the interface for data. It is in charge of data storage. It is a database that represents the logical data structure that underpins the entire application.

**Django channels:** It will allow to handle web sockets, chat protocols. Channels is an integration layer that lets us manage other connections in a synchronous or asynchronous fashion.

**Redis server:** Redis is primarily used as a database for storing user/message data and for communicating across connected servers. Redis includes Pub/Sub messaging, enabling developers to extend the backend by generating many server instances. It can handle a wide range of rich data structures, including Lists, Sets, Sorted Sets, Hashes, and so on.

**Cron job:** It is a utility that allows us to plan scripts or commands to execute at a specific time, date, or interval, and these tasks or jobs are referred to as "cron jobs". In our project, we use it for periodical backup of users’ data and verifying subscription plans.

1. Written requirements specifications (more information and a template will be given and discussed in class):
   1. All functional requirements
   2. All non-functional requirements
   3. Interfaces (user, hardware, software, and/or communication)

The requirements should be detailed enough to build the project.

All the requirement specifications are written in other document in SRS template format.

1. The plan for implementing the project through three development phases. For each development phase, the corresponding requirements should be identified (need to prioritize functionalities based on their criticality).

**Project Implementation Plan:**

A software project implementation plan is an important checklist that is created prior to the implementation of a user story. To put it another way, it's a set of step-by-step instructions prepared by an engineer for himself or any other engineer to follow in order to complete a task. Our project provides a separate Dashboard based on users like Admin, Designer, User and their functionalities. The following are the tasks chosen based on our customer's requirements that in each module, we will implement and accomplished all the tasks.

**Phase 1:**

**Admin Dashboard:**

It is a platform for administrators that let them view and manipulate data through the website user interface. User-related duties such as offering insight into user activity, dealing with profiles that breach the website terms and conditions, and tracking transactions can all be aided by the admin panel. However, we should be aware that admin sites are not solely for user-related duties.

* **Task 1:** Admin should have his credentials to log into the application like their email id and password are need to be entered while logging into the application.
* **Task 2:** Admin should be given access to add or remove designers as he is responsible for managing the application or responsible for customer user experience.
* **Task 3:** Admin should have access to manage subscriptions and add plans as we are providing plans for users to choose from based on how long they need subscriptions for our application. The plans that they will be able to choose are silver for a three-month subscription, gold for six months and diamond for 12 months.
* **Task 4:** Admin can view users/designer profiles to know details like status or the plan chosen for the subscription.

**Phase 2:**

**Designer Dashboard:**

It's a platform dedicated to designers, where they can try out and use our application. The Designer Dashboard's functionality and features are subject to constraints. Designers, unlike administrators, are unable to add or remove people or manage plans. The functions that Designer is capable of being entirely dependent on their requirements. The following are some of the features available to Designer's Dashboard users.

* **Task 1:** New Designers should be provided with an option to register into our application by filling in their personal details like email id, password and more. After successful registration, they will obtain their credential to use for login to the application
* **Task 2:** After successful registration, the page should redirect to the login page for logging into the Dashboard using their credentials
* **Task 3:** Designers should also have access to view user’s profiles so they can connect and have conversations with them.
* **Task 4:** Designers should be given an option to change their password in case of emergency
* **Task 5:** To connect with users, Designers will be given a chat option to interact with users.

**Phase 3:**

**User Dashboard:**

The user dashboard gives the user a comprehensive picture of the system, allowing them to access the most critical data, operations, and controls. In practice, a dashboard frequently serves as a homepage. This Dashboard is designed for users who wish to use our application. User Dashboard functionalities are different when compared with other Dashboards. Here we help users choose the Designer of their choice. Users will not be given access to add or remove users or add plans. The features that users can experience are listed below.

* **Task 1:** New users are allowed to register into the application by clicking on the register button on the login page where they redirect to registration for requesting details to be filled like email id password and more.
* **Task 2:** Users should be able to view Designers profiles to know their past work.
* **Task 3:** Users are given an option to chat with the designers they like to know more about their work.
* **Task 4:** User are allowed to give feedback to the designers based on conversation.
* **Task 5:** Users have access to view their details and can verify them.
* **Task 6:** Users can also change the password if there is an emergency

1. Member contribution table (should describe who wrote what parts of the report). Add more rows as needed.

|  |  |  |  |
| --- | --- | --- | --- |
| Member name | Contribution description | Overall Contribution (%) | Note  (If applicable) |
| Sravanthi | Functional Requirements, Member contribution table | 12.5 |  |
| Manish | Functional Requirements | 12.5 |  |
| Manohar | Non functional  Requirements | 12.5 |  |
| Sharath | Interface Requirements | 12.5 |  |
| Teja | Structure of the System | 12.5 |  |
| Harsha | Description of the individual components | 12.5 |  |
| Praveen | Implementation of the projects in phases | 12.5 |  |
| Yeshwanth | Implementation of the projects in phases | 12.5 |  |

1. The updated meeting minutes (in the project repository).

**NOTE**: Submit one document (e.g., Deliverable-2.pdf) that contains 1, 2, 3, and 4.

**Document formats:** Your documents may be in any of the following formats (1) MS Word (2) Adobe Acrobat (pdf). If you plan to use other formats, please discuss this with the instructor to confirm compatibility.

**Grading:** You will be graded on how well you have completed each of the required elements and on your adherence to the directions given. Credit will be given for the completeness and level of detail placed into each of the required elements.

**Group Policy:** Each team member will receive the same grade for their assignment, however, if particular members are not participating, adjustments to that individual’s grade may be made.

**What to Hand In:** At 11:59pm on the due date the instructor or TAs will take a snapshot of each of your directories and use this for grading. Create a “**Note-deliverable-2**” file (a top level, plain text) to indicate the names of deliverable files and their locations for this deliverable.