Frontend: React.js (SPA)

- Technology: React.js
- Description: The frontend is built as a Single Page Application (SPA) to provide a smooth user experience with dynamic content updates without the need to reload the page.
- Key Features:
    - Component-based architecture
    - Virtual DOM for efficient rendering
    - React Router for managing navigation
    - State management using Redux or Context API
    - Responsive design using CSS frameworks like Bootstrap or Material-UI

Backend: Django (Python)

- Technology: Django
- Description: The backend is developed using Django, a high-level Python web framework that encourages rapid development and clean, pragmatic design.
- Key Features:
    - Django Rest Framework (DRF) for building robust RESTful APIs.
    - Authentication and Authorization mechanisms.
    - Integration with MongoDB using packages like Djongo or MongoEngine.
    - Scalability and Security features.

Database: NoSQL database using MongoDB

- Technology: MongoDB
- Description: A NoSQL database is used for storing data, offering flexibility in schema design and scalability.
- Key Features:
    - Document-oriented storage (BSON format)
    - High performance for read/write operations
    - Sharding for horizontal scalability
    - Indexing for efficient querying

Architectural Design

1. Component Diagram

```
+-------------------------------------------------+
|                    Frontend (React.js)          |
|                                                 |
| +----------------------+   +----------------+   |
| |  Pages (SPA)         |   | Components     |   |
| | - Index              |   | - Navbar       |   |
| | - Register           |   | - Image Slider |   |
| | - Login              |   | - Forms        |   |
| | - Dashboard          |   | - Notifications|   |
| | - Inbox              |   | - Chat         |   |
| | - Assignments        |   +----------------+   |
| | - Projects           |                        |
| | - Manage Projects    |                        |
| | - Transactions       |                        |
| +----------------------+                        |
+-------------------------------------------------+

       | RESTful API calls (HTTP/HTTPS)

+-------------------------------------------------+
|                     Backend (Django)            |
|                                                 |
| +------------------+    +-------------------+   |
| | Django Framework |    | Django REST       |   |
| | - Models         |    | Framework (DRF)   |   |
| | - Views          |    | - API Endpoints   |   |
| | - Templates      |    | - Serializers     |   |
| | - URLs           |    +-------------------+   |
| +------------------+                            |
|                                                 |
+-------------------------------------------------+

       | Database queries

+-------------------------------------------------+
|                     Database (MongoDB)          |
|                                                 |
| +----------------+   +----------------------+   |
| | Collections    |   | Document Structure   |   |
| | - Users        |   | - User Profile       |   |
| | - Projects     |   | - Project Details    |   |
| | - Assignments  |   | - Assignment Details |   |
| | - Chats        |   | - Chat Messages      |   |
| | - Transactions |   | - Transaction Records|   |
| +----------------+   +----------------------+   |
|                                                 |
+-------------------------------------------------+
```

2. **Data Flow**
   1. **User Interaction**
      - Users interact with the frontend SPA via their web browsers.
      - React components handle user inputs and actions.
   2. **Frontend to Backend Communication**
      - React.js communicates with the Django backend via RESTful API calls.
      - HTTP/HTTPS protocols are used for secure communication.
   3. **Backend Processing**
      - Django processes incoming API requests.
      - Business logic is executed, and necessary database operations are performed.
   4. **Database Operations**
      - Django interacts with MongoDB to fetch, insert, update, or delete data.
      - MongoDB returns the processed data to the Django backend.
   5. **Backend to Frontend Response**
      - Processed data is sent back to the React.js frontend via RESTful API responses.
      - React components update the UI based on the received data.

3. **Detailed Component Description Frontend (React.js):** The frontend of the application is built using React.js, a popular JavaScript library for building user interfaces. The application follows the Single Page Application (SPA) structure, ensuring a smooth and interactive user experience without frequent page reloads. Below is an expanded description of the components and their roles within the application.

Pages (SPA)

The SPA structure is designed to handle different views efficiently. Each page represents a different functional area of the application:

- **Index:** The landing page of the application that provides an overview of features, a brief introduction, and testimonials.
- **Register:** A multi-step registration process for new users to sign up.
- Login: Allows existing users to authenticate and access their accounts.
- **Dashboard:** The main user interface where users can view notifications, assignments, projects, and other relevant information.
- **Inbox:** A communication hub where users can chat with team members and receive notifications.
- **Assignments:** Displays all assignments, with tabs for due, completed, and all assignments.
- **Projects:** Lists all projects with tools for searching, sorting, and filtering. Users can view detailed project information and accept invites.
- **Manage Projects:** Provides tools for project hosts to manage their projects, team members, assignments, and project status.
- **Transactions:** Shows a list of all transactions made or received by the user.

Components

Navbar

- **Description:** Provides navigation links to the different pages of the application.
- **Key Features:**
  - ➔ **Navigation Links:** Includes links to Inbox, Assignments, Projects, Manage Projects, and Transactions.
  - ➔ **Dropdown Menus:** Offers additional links like About Us, Testimonials, and Docs under the More section.
  - ➔ **User Authentication:** Displays Login/Register options when the user is not logged in and user profile options when the user is authenticated.

Image Slider

- **Description:** Displays feature highlights in a visually appealing carousel format.
- **Key Features:**
  - ➔ **Slides:** Each slide can display different feature information, including images and text.
  - ➔ **Auto-Play:** Automatically transitions between slides with an adjustable interval.
  - ➔ **Manual Control:** Allows users to manually navigate between slides using navigation arrows or dots.

Forms

- **Description:** Handles user input for various functionalities, including registration and login.
- **Key Features:**
  - ➔ **Input Fields:** Various form fields for capturing user data such as name, email, password, address, etc.
  - ➔ **Validation:** Client-side validation to ensure data integrity before submission.
  - ➔ **Multi-Step:** Supports multi-step forms for processes like registration (divided into different pages for better UX).
  - ➔ **Buttons:** Includes buttons for form navigation (Back, Next, Register, Login).

- ➔

Form Examples

1. **Registration Form:**
   Step 1: Collects basic user information (First Name, Last Name, Date of Birth, Gender, Email, Password, Confirm Password).
   Step 2: Collects address details (Flat/House No., Apartment, Street, Landmark, State, City, Pin-code, Country).
   Step 3: Collects contact information (Country, Mobile No., OTP verification).
   Step 4: Collects identification information (Aadhar No., OTP verification, Captcha).
2. **Login Form:**
   Step 1: Collects user credentials (Email/Mobile No., Password) with options for Remember Me and Forgot Password.
   Step 2: Performs OTP verification and captcha.

Notifications

- Description: Displays alerts and updates to the user.
- Key Features:
- Popup Notifications: Shows real-time alerts for new messages, project updates, assignment deadlines, etc.
- Notification Center: Aggregates all notifications in a single view for easy access.
- User Interaction: Allows users to mark notifications as read, dismiss them, or navigate to related content.

Additional Components

Footer

- **Description:** Provides links to additional information and resources.
- **Key Features:**
  - ➔ **Links:** Includes links to the About Us page, Contact Us, Privacy Policy, Terms of Service, and other relevant pages.
  - ➔ **Social Media Icons:** Links to social media profiles of the application.

Sidebar

- **Description:** Provides additional navigation or functional tools for users.
- **Key Features:**
  - ➔ **Quick Links:** Links to frequently accessed pages or actions.
  - ➔ **Widgets:** Displays additional information like recent activities, quick stats, etc.

Modals

- **Description:** Used for displaying important information or capturing user input without navigating away from the current page.
- **Key Features:**
  - ➔ **Popups:** Appears as overlays to provide contextual information or actions (e.g., viewing project details, confirming actions).
  - ➔ **Forms:** Can include forms for quick actions like sending messages or updating profile details.