BOSTON UNIVERSITY

Software Design Document (SDD - Team1)

CS-673 (Software Engineering)

Fall 2024

By

Sergio Khalil

Yuhang Zhang

Asma Asiri

Brad Nissenbaum

Shivaang Kumar

Table of Contents

1. Introduction **4**
- 1.1 Purpose 4
- 1.2 Scope 4
- 1.3 Definitions 4
- 1.4 Acronyms/Abbreviations 5

2. References **5**

3. Decomposition Description **6**
- 3.1 Module decomposition 6
  - 3.1.1 Authentication and Authorization Module 6
  - 3.1.2 Checklist Management Module 6
  - 3.1.3 Document Management Module 6
  - 3.1.4 Notification Module 6
  - 3.1.5 Dashboard and Reporting Module 6
  - 3.1.6 Third-Party Integration Module 6
- 3.2 Concurrent process decomposition 7
  - 3.2.1 Real-Time Notification Process 7
  - 3.2.2 Data Synchronization Process 7
- 3.3 Data decomposition 7
  - 3.3.1 User Entity 7
  - 3.3.2 Checklist Entity 7
  - 3.3.3 Document Entity 7
  - 3.3.4 Notification Entity 8

4. Dependency Description **8**
- 4.1 Intermodule dependencies 8
- 4.2 Interprocess dependencies 8
- 4.3 Data dependencies 9

5. Interface Description **9**
- 5.1 Module interface 9
  - 5.1.1 Authentication and Authorization Module Interface 9
  - 5.1.2 Checklist Management Module Interface 10
  - 5.1.3 Document Management Module Interface 10
  - 5.1.4 Notification Module Interface 10
  - 5.1.5 Third-Party Integration Module Interface 11
- 5.2 Process interface 11
  - 5.2.1 Real-Time Notification Process Interface 11
  - 5.2.2 Data Synchronization Process Interface 11

6. Detailed Design **12**

6.1 Module detailed design 12
    6.1.1 Authentication and Authorization Module 12
    6.1.2 Checklist Management Module 12
    6.1.3 Document Management Module 13
    6.1.4 Notification Module 13
    6.1.5 Third-Party Integration Module 13
6.2 Data detailed design 14
    6.2.1 User Entity 14
    6.2.3 Document Entity 15
    6.2.4 Notification Entity 15

1. Introduction

1.1 Purpose

The Real Estate Checklist Tracker project aims to create a comprehensive tool for managing the leasing process. This tool will foster better communication between real estate agents and their clients. It will enable agents to track required documents, deadlines, and the progress of leasing applications efficiently, ensuring transparency, organization, and reduced manual paperwork throughout the process.

1.2 Scope

The Real Estate Checklist Tracker is designed to support the entire leasing application process by providing functionalities such as:

- **Checklist Creation and Tracking**: Customizable checklists to accommodate various client needs.
- **Real-Time Notifications**: Automated alerts for completed tasks and upcoming steps to ensure timely action.
- **Document Management**: Secure upload, storage, and management of documents.
- **Dashboard Views**: Brokers and agents will have access to dashboards displaying leasing progress, facilitating better decision-making.
- **Third-Party Integration**: This feature enables agents to run and store credit and background checks within the platform, streamlining leasing decisions.

This platform will cater to brokers, real estate agents, and clients, providing tailored access and functionalities based on their roles.

1.3 Definitions

- **Checklist**: A customizable list of tasks and document requirements for the leasing application.
- **Dashboard**: An interface providing users with an overview of tasks and application progress.
- **GUI (Graphical User Interface)**: The platform's interactive visual components facilitate user interactions.
- **API (Application Programming Interface)**: A set of protocols for integrating third-party services like credit checks.
- **Real Estate Agent**: A professional who manages leasing and interacts directly with clients.
- **Broker**: A manager overseeing agents and their clients' progress within the leasing process.
- **Client**: An individual seeking to lease a property through an agent.
- **Socket.io**: A technology enabling real-time, bidirectional, event-based communication.

1.4 Acronyms/Abbreviations

- **API** - Application Programming Interface
- **SRS** - Software Requirements Specification
- **SDD** - Software Design Document
- **SCMP** - Software Configuration Management Plan
- **SPMP** - Software Project Management Plan
- **GUI** - Graphical User Interface
- **JIRA** - A project management tool for Agile workflows

2. References

1. Software Requirements Specification (SRS) - Defines the functional and non-functional requirements for the Real Estate Checklist Tracker, covering essential use cases, user roles, and high-level system functionality.

2. Software Project Management Plan (SPMP): This plan outlines the project management approach, including methodologies, team roles, timelines, and risk management strategies, applied to this project.

3. Software Configuration Management Plan (SCMP): This plan Details configuration management practices, such as version control, change management, and release processes, for the project's codebase, documentation, and resources.

4. Draft Proposal: This overviews the project's objectives, benefits, targeted user roles, and critical functionalities envisioned for the Real Estate Checklist Tracker.

5. IEEE 1016-1998 Standard for Software Design Descriptions—This is a guideline for structuring this SDD document, ensuring best practices in software design documentation.

6. Tools and Technologies:

Jira - Project and issue management tool used for Agile workflows, tracking epics, user stories, and tasks throughout the project.

GitHub - Source code repository and version control platform, facilitating collaborative development and code reviews.

Figma - Design tool for UI/UX prototyping, used for creating wireframes and interactive mockups for the GUI.

Each reference serves as a foundational element to ensure the Real Estate Checklist Tracker's design, implementation, and management meets project goals and adheres to industry standards.

3. Decomposition Description

3.1 Module decomposition

The Real Estate Checklist Tracker system is divided into several main modules, each responsible for specific functionality. These modules work together to provide a seamless experience for brokers, agents, and clients.

3.1.1 Authentication and Authorization Module

- **Description:** Manages secure login, registration, and access control for different user roles (e.g., Brokers, Agents, Clients). It includes role-based access control (RBAC) to enforce data privacy and restrict features based on user type.

3.1.2 Checklist Management Module

- **Description**: Allows agents to create, customize, and manage client checklists. This module tracks task completion, sets deadlines, and updates checklist items based on the leasing process.

3.1.3 Document Management Module

- **Description**: Facilitates secure uploading, storing, and accessing of documents by clients and agents. This module validates file types and sizes and organizes documents for easy retrieval.

3.1.4 Notification Module

- **Description**: Sends real-time notifications and reminders to clients and agents regarding checklist updates, task deadlines, and document uploads. It supports in-app alerts as well as email notifications for timely updates.

3.1.5 Dashboard and Reporting Module

- **Description**: Provides a visual overview of leasing progress for agents and brokers, allowing them to view client status, track checklist completion, and access real-time data on pending and completed tasks.

3.1.6 Third-Party Integration Module

- **Description**: Integrates with external services, such as credit and background check providers, to streamline the verification process within the platform. This module retrieves results and uploads them for agent review

3.2 Concurrent process decomposition

The Real Estate Checklist Tracker system operates multiple processes concurrently to ensure a smooth and responsive user experience:

3.2.1 Real-Time Notification Process

Description: A background process that continuously listens for checklist updates, task completions, and other triggers, sending real-time alerts to relevant users (e.g., clients and agents). This process ensures timely updates using Socket.io for push notifications.

3.2.2 Data Synchronization Process

- **Description**: Maintains data consistency across the document management system, checklist status, and dashboards. This process ensures that updates made by one user are immediately visible to others, especially in high-traffic situations with concurrent users.

3.3 Data decomposition

The system relies on structured data entities to support the leasing process and track user interactions efficiently:

3.3.1 User Entity

Description: Represents user types (Brokers, Agents, Clients) with associated attributes such as user ID, role, access permissions, and contact information. It supports authentication and role-based access control .

3.3.2 Checklist Entity

Description: Contains information on each client's leasing requirements, including task names, descriptions, deadlines, and completion status. This entity links to the client and agent for easy tracking and management.

3.3.3 Document Entity

Description: Stores metadata for documents uploaded by clients and agents, including file name, type, upload timestamp, and status. This data entity supports secure file handling and version control.

3.3.4 Notification Entity

Description: Logs and manages notifications, including notification type, recipient, timestamp, and read/unread status. It allows users to track important events within the leasing process.

Each decomposition layer contributes to the system's modularity, allowing for efficient development, maintenance, and scalability while maintaining a seamless user experience.

4. Dependency Description

4.1 Intermodule dependencies

Authentication and Authorization Module ↔ Checklist Management Module

- The Checklist Management Module relies on the Authentication and Authorization Module to verify user identity and permissions, ensuring that only authorized agents and clients can access or modify checklists.

Checklist Management Module ↔ Document Management Module

- The Checklist Management Module depends on the Document Management Module to handle document uploads for specific checklist items. When a client completes a checklist task by uploading a required document, this dependency ensures seamless integration between task management and document handling.

Notification Module ↔ All Other Modules

- The Notification Module depends on various modules (Checklist, Document Management, and Authentication) to trigger alerts for task updates, document submissions, and role-based access events. This ensures users are promptly informed of any system changes or required actions.

Third-Party Integration Module ↔ Checklist and Document Management Modules

- The Third-Party Integration Module relies on the Checklist and Document Management Modules to store and display external verification results (e.g., credit checks). This dependency ensures all relevant documentation is available within the checklist for agent review and client transparency.

4.2 Interprocess dependencies

Real-Time Notification Process ↔ Data Synchronization Process

- The Real-Time Notification Process depends on the Data Synchronization Process to ensure that updates to checklists, documents, and user actions are promptly reflected across all user interfaces. This dependency minimizes discrepancies in data seen by users interacting with the system simultaneously.

Data Synchronization Process ↔ Document and Checklist Updates

- The Data Synchronization Process depends on timely updates from the Document and Checklist Management Modules to ensure consistent status displays on dashboards. For example, when a client uploads a document, the synchronization process ensures that the update is immediately reflected on the agent's dashboard.

4.3 Data dependencies

User Entity ↔ Checklist and Document Entities

- User data (roles and permissions) is essential for accessing and modifying checklists and documents. For instance, only authorized agents and clients can upload or view specific documents based on their user role.

Checklist Entity ↔ Document and Notification Entities

- Each checklist item may require associated documents, establishing a dependency between the Checklist and Document Entities. Additionally, changes to checklist items (e.g., task completion) trigger notifications, creating a dependency between Checklist and Notification Entities

Notification Entity ↔ User and Checklist Entities

- The Notification Entity relies on data from the User and Checklist Entities to determine each notification's correct recipients and content. For example, when a client uploads a document, the associated agent receives a notification based on the checklist and user data.

5. Interface Description

5.1 Module interface

Each Real Estate Checklist Tracker system module has defined interfaces facilitating interaction with other modules, external services, or users.

5.1.1 Authentication and Authorization Module Interface

- **Interfaces with**: All modules.

- **Description**: This module provides a login interface for brokers, agents, and clients. It validates user credentials and enforces role-based access control (RBAC). The module also interfaces with other modules by passing user roles and permissions to restrict or allow access as needed.
- **Methods**:
  - Login (username, password): Validates user credentials.
  - Authorize (user_id, role): Grants or denies access based on role.
  - Logout (user_id): Terminates user session.

5.1.2 Checklist Management Module Interface

- **Interfaces with** Document Management, Notification, and Dashboard.
- **Description**: This interface allows agents to create, update, and monitor checklist tasks for clients. It retrieves user permissions from the Authentication Module and interacts with the Document Management Module to handle document uploads. The interface updates the notification module when checklist tasks are modified or completed.
- **Methods**:
  - createChecklist(client_id, tasks): Initializes a new checklist for a client.
  - updateChecklist(task_id, status): Updates the status of a specific task.
  - retrieveChecklist(client_id): Fetches the checklist associated with a client.

5.1.3 Document Management Module Interface

- **Interfaces with** Checklist Management, Dashboard, Notification, and Third-Party Integration.
- **Description**: Handles file uploads and retrievals, supporting secure document storage and access for clients and agents. It interfaces with the Checklist Management Module to link documents to specific tasks and sends upload notifications to the Notification Module.
- **Methods**:
  - uploadDocument(client_id, file): Validates and uploads a document to the system.
  - getDocumentStatus(document_id): Retrieves the current status of a document.
  - linkDocumentToChecklist(task_id, document_id): Associates an uploaded document with a checklist task.

5.1.4 Notification Module Interface

- **Interfaces with**: All modules.

- **Description**: The Notification Module interface manages real-time and scheduled notifications using the Checklist, Document, and Authentication Modules triggers. It delivers notifications via in-app alerts or email.
- **Methods**:
  - sendNotification(user_id, message): Sends a custom notification to a user.
  - getNotificationHistory(user_id): Retrieves recent notifications for a specific user.

5.1.5 Third-Party Integration Module Interface

- **Interfaces with** Checklist Management and Document Management.
- **Description**: This module integrates with external services for background and credit checks. It provides an API to request and store external data and associates it with the client's profile within the Document Management Module.
- **Methods**:
  - requestCreditCheck(client_id): Sends a credit check request to an external provider.
  - retrieveReport(report_id): Fetches the latest report from an external provider.

5.2 Process interface

Process interfaces in the system ensure that concurrent processes like real-time notifications and data synchronization run smoothly across modules.

5.2.1 Real-Time Notification Process Interface

- **Interfaces with** Notification, Checklist, and Document Management.
- **Description**: This process listens for checklist or document status changes and sends immediate alerts to users. It triggers notifications based on checklist updates, document uploads, and deadline reminders.
- **Methods**:
  - triggerAlert (event, user_id): This sends an alert for a specific event (e.g., an uploaded document).
  - Schedule reminder (task_id, due_date): Schedules a reminder notification for a checklist task.

5.2.2 Data Synchronization Process Interface

- **Interfaces with** Document Management, Checklist Management, and Dashboard.

- **Description**: Ensures that updates to checklists and documents are reflected in real-time across the dashboard and other modules. This process regularly synchronizes data, mainly when multiple users are active concurrently.
- **Methods**:
  - syncChecklistData(client_id): Syncs checklist data for a client across modules.
  - syncDocumentData(document_id): Syncs document status and metadata across modules.

6. Detailed Design

Each Real Estate Checklist Tracker system module has specific internal components and logic that contribute to its functionality. The following provides a detailed breakdown of each core module.

6.1 Module detailed design

6.1.1 Authentication and Authorization Module

- **Components**:
  - **Login Controller**: Manages user login requests, validates credentials, and initiates sessions.
  - **Role-based access Control (RBAC)** Implements permissions based on user roles (Broker, Agent, Client).
  - **Session Manager**: Manages user sessions, including login, logout, and session expiration.
- **Logic**:
  - Upon login, credentials are validated, and if successful, a session is created, and user roles/permissions are assigned.
  - Access to specific features and data is dynamically controlled based on user role, ensuring data security and integrity across all system interactions.

6.1.2 Checklist Management Module

- **Components**:
  - **Checklist Controller**: Allows agents to create, update, and delete checklists for each client.
  - **Task Manager**: Handles individual checklist tasks, tracking status, deadlines, and task completion.
  - **Client Tracker** Links each checklist to the client's profile, ensuring the data remains specific to the user.
- **Logic**:
  - Agents can initialize checklists with customized tasks and deadlines. Task status is dynamically updated based on user interactions and changes trigger notifications.

    - Dependencies on other modules ensure that document uploads are linked to tasks and progress is displayed on the user's dashboard.

6.1.3 Document Management Module

- **Components**:
    - **File Uploader**: Facilitates secure document uploads, validating file type and size.
    - **Storage Handler**: Organizes uploaded documents in a secure repository with unique identifiers and version control.
    - **Access Controller**: Manages document permissions, ensuring only authorized users (based on role) can view or modify documents.
- **Logic**:
    - Documents are uploaded, validated, and securely stored. Each document is associated with a specific checklist task and client. Version control allows for document replacement without data loss.
    - Document updates trigger notifications, and access to each document is controlled based on the user's role (e.g., agents can view all client documents; clients can view only their own).

6.1.4 Notification Module

- **Components**:
    - **Notification Trigger**: Monitors events across modules (e.g., task updates and document uploads) to initiate notifications.
    - **Notification Manager**: Controls the format and delivery of notifications (in-app alerts and emails).
    - **Notification Log**: Maintains a history of sent notifications for audit and accountability purposes.
- **Logic**:
    - Changes within the system, such as completed tasks or document uploads, trigger notifications. Each notification is formatted and sent according to user preference, and a record is kept in the notification log for future reference.

6.1.5 Third-Party Integration Module

- **Components**:
    - **API Connector**: Manages communication with external services, including initiating and retrieving background and credit check reports.
    - **Report Manager**: Stores and links external reports to the client profile and checklist.
- **Logic**:
    - API requests are made for credit or background checks, and responses are stored within the system for agent review. If errors occur, notifications are sent to prompt manual retries or further action.

6.2 Data detailed design

The data entities within the Real Estate Checklist Tracker system store essential information for user roles, tasks, documents, and notifications. Each entity's design ensures data consistency and efficient access control.

6.2.1 User Entity

- **Attributes:**
  - user_id: Unique identifier for each user.
  - Role: Defines user role (Broker, Agent, Client).
  - Credentials: Stores encrypted login details.
  - Permissions: Stores RBAC-specific permissions.
- **Logic:**
  - This entity is critical for authenticating users, defining access rights, and maintaining secure session information for every interaction within the system.

6.2.2 Checklist Entity

- **Attributes**:
  - checklist_id: Unique identifier for each checklist.
  - client_id: Links the checklist to a specific client.
  - Tasks: Array of functions, each with details (name, status, deadline).
- **Logic**:
  - Each checklist is created and modified based on user input. Individual tasks are updated according to document submissions and deadline completions. This entity also tracks task completion percentages displayed on the client and agent dashboards.

6.2.3 Document Entity

- **Attributes**:
  - document_id: Unique identifier for each document.
  - file_path: Path where the document is stored.
  - upload_timestamp: Time of document upload.
  - Status: Tracks document status (e.g., Pending, Approved).
- **Logic**:
  - Document attributes ensure secure storage and controlled access. The status attribute is updated based on user actions and checklist requirements, with linked notifications sent to the relevant users.

6.2.4 Notification Entity

- **Attributes**:
  - notification_id: Unique identifier for each notification.
  - recipient_id: User receiving the notification.
  - Message: Notification content.
  - Timestamp: When the notification was sent.
  - Status: Indicates if the notification is read or unread.
- **Logic**:
  - This entity stores and tracks notifications and logs details about each one. It ensures a consistent record of alerts for user actions and system events.