MILESTONE 1 Report

CS396 Software Design Principles and Practices

People: Al Nahiyan and Saeed AlKaabi

Option

Option 1 - Design Your Own Product

Link to Project GitHub Repository

https://github.com/nu-cs-sw-design/project-20252601-396fp

Project Planning

Week 1: Requirements and Initial Design

- Define complete use cases and user stories
- Create initial architectural design considerations
- Identify quality attributes and priorities
- Submit Milestone 1 report

Week 2: Implementation of Messy Prototype

- Implement core functionality (user authentication, item listing, rental requests)
- Create basic UI/UX for primary workflows
- Focus on getting features working (prioritize functionality over design)
- Submit Milestone 2 progress report

Week 3: Design Analysis and Refinement

- Analyze messy code for design principle violations
- Create comprehensive class diagrams
- Document architectural decisions
- Identify refactoring opportunities
- Prepare final deliverables and submit

Initial Analysis

Purpose

The Campus Rental Platform enables students to participate in a peer-to-peer rental marketplace exclusively within their academic community. The platform addresses the financial constraints students face by providing an affordable alternative to traditional rental services through direct student-to-student transactions.

Key Value Propositions:

- **For Rentees (Borrowers):** Access items at significantly reduced costs compared to commercial rental services, with convenient campus-based pickup and return
- **For Renters (Lenders):** Generate supplementary income from underutilized possessions while helping fellow students
- **For the Community:** Foster a sustainable, collaborative economy that builds trust and connections within the student body

Product Description

The Campus Rental Platform is a mobile-first web application that facilitates secure, time-bound rentals between verified students. The system manages the complete rental lifecycle from item discovery to payment processing and dispute resolution.

Target Users

Primary Users:

1. **Full-Time Campus Students** - Students enrolled in on-campus programs with regular access to campus facilities
2. **Part-Time Campus Students** - Students with limited campus presence who need occasional access to items

User Constraints:

- Must have valid .edu email address for verification
- Must be currently enrolled students
- Must have access to campus for item exchange

Core Product Features

1. User Authentication and Verification System

- Dual email registration (primary contact email + university .edu email)
- Student status verification through university email confirmation
- Secure login with password recovery

2. Item Listing Management

- Create detailed rental listings with photos, descriptions, and pricing
- Set availability calendars and rental durations (hourly, daily, weekly)
- Define pickup/return locations (campus locations, dorm buildings, etc.)
- Categorize items (electronics, textbooks, sports equipment, tools, etc.)
- Mark items as unavailable or remove listings

3. Discovery and Search System

- Browse available items by category

- Filter by: price range, location, availability dates, item condition, renter ratings
- AI-powered recommendation engine matching rentee searches to relevant listings
- Interactive campus map showing item locations and proximity
- Save favorite items and create wish lists

4. Rental Request and Booking Flow

- Request to rent with proposed dates and times
- Calendar integration showing availability
- Rental agreement generation with terms and conditions
- Renter approval/denial with optional messaging
- Automated booking confirmations (Maybe use third party calendar tools like Calendly or Google Calendar)

5. Payment Processing

- Integrated payment gateway supporting multiple methods (credit/debit, digital wallets, digital services like Adyen)
- Security deposit handling (held during rental period)
- Automatic payment release after successful return
- Transaction history and receipt generation
- Refund processing for cancellations

6. Communication System

- In-app messaging between renters and rentees
- Automated notifications for:
    - New rental requests
    - Booking confirmations/denials
    - Payment confirmations
    - Pickup reminders
    - Return reminders
    - Late return warnings
- Real-time chat for coordination
- Message history and archiving

7. Reputation and Trust System

- Two-way rating system (renters rate rentees, rentees rate renters)
- Written reviews with timestamps

- Reputation scores affecting search rankings
- Badge system for reliable users (verified, top renter, responsible rentee)
- Report functionality for inappropriate behavior

8. Scheduling and Logistics

- Pickup location coordination (with map pinning)
- Proposed pickup times with confirmation workflow
- Return scheduling with automated reminders
- Extension requests during active rentals

9. User Profile and Dashboard

- View rental history (as renter and rentee)
- Track active rentals
- Manage listings
- Update payment methods
- Adjust notification preferences
- View earnings and spending analytics

Future considerations:

Dispute Resolution Framework

- Report damaged or non-returned items
- Photo evidence upload
- Admin review system (future phase)
- Insurance claim integration (future phase)

Use Cases

UC-1: User Registration & Verification

System Description: The Registration System allows new students to create an account and verify their university affiliation using a .edu email. The system ensures that only legitimate campus users gain access to the platform.

Actor: New Student User

- A person who wants to create an account in order to use the rental platform.

Actor's Goal:

- To successfully register and activate a verified student account so they can access all platform features.

Main Flow:

1. User opens the Registration Page.
2. User enters primary email, password, personal information, and .edu email.
   a. If the .edu email entered is not valid:
      I. System requests the user to enter a valid university email
      II. User corrects the email and continues the flow
3. User submits the registration form
4. System displays the "Check Your Email" confirmation page.
5. User opens the verification email and clicks the verification link.
   a. if the user clicks an expired link:
      I. System informs the user that the link has expired
      II. System provides a "Resend Verification Link" option
      III. User clicks the new link when it arrives
   b. If the user does not receive the email:
      I. User clicks "Resend Email" on the "Check Your Email" Page
      II.System sends a new verification email
6. System verifies the link and activates the account.
7. User is redirected to the Login Page.

Preconditions

- A verified student account is created
- The user is allowed to log in and access protected features.

UC-2: Create Rental Listing

System Description: The Listing Management System allows verified student renters to create item listings with photos, details, and availability. Once submitted, listings become visible to other students searching for rentals.

Actor: Student Renter

- A verified user who wants to post an item for rent on the platform.

Actor's Goal:

- To successfully create and publish a rental listing that other students can view and request.

Main Flow:

1. User opens the Create Listing Page.
2. User uploads one or more photos.
    a. If the file is too large:
        i. The system notifies the user that the photo file is too large.
        ii. User uploads a smaller image.
3. User enters listing details, including title, description, category, and price.
4. User sets availability dates and pickup location.
    a. Date selected is in the past:
        i. The system notifies the user that the selected date is in the past.
        ii. User selects a valid future date
5. User submits the listing form
    a. Dates Conflict With Existing Availability
        i. System notifies the user that the availability range conflicts with existing date settings or is inconsistent
        ii. User corrects the dates
6. System validates all fields.
7. System publishes the listing.
8. Listing becomes available on the Search/Browse Pages.

Preconditions

- Listing becomes publicly visible to all users.
- Listing appears under the user's My Listings Page

UC-3: Search and Request Item Rental

System Description: The Search and Discovery System allows verified students to browse items, apply filters, view item details, select rental dates, and submit rental requests. The system ensures accurate date availability and cost calculation.

Actor: Student Rentee

- A verified user looking to search for and request an item for rent.

Actor's Goal:

- To find an item, review its details, select available dates, and submit a rental request.

Main Flow:

1. User opens the Search Page or Browse Page.
2. User enters search terms or browses categories.
    a. If no items match:
        i. System suggests adjusting or broadening filters.
        ii. User adjusts filters or search terms.
3. System displays matching items with filter options.
4. User selects an item and views its Item Details Page.
5. User selects desired rental dates.
    a. If dates are in the past:
        i. System notifies the user that past dates cannot be selected.
        ii. User selects valid future dates
    b. If dates are unavailable:
        i. System suggests alternative dates.
        ii. User selects valid available dates.
6. System calculates the estimated rental cost.
7. User submits a rental request.
8. System creates the request and notifies the renter.

Preconditions

- A pending rental request is created.
- Renter is notified of the request.

UC-4: Approve or Deny Rental Request

System Description: The Rental Request Management System allows item owners to review incoming requests and choose to approve or deny them. Approved requests proceed to payment.

Actor: Student Rentee

- The owner of the item receiving the request.

Actor's Goal:

- To review rental details and either approve or deny the request.

Main Flow:

1. User opens the Requests Page.
2. User selects a pending rental request.
3. User reviews dates, price, and rentee's profile.
4. User chooses to approve or deny the request.
    a. If Approved:
        i. System notifies the rentee.
        ii. Rentee is redirected to the Payment Page.
        iii. Rentee completes payment.
            1. If payment fails:
                a. The request remains pending until payment succeeds.
        iv. System confirms the rental booking.
    b. If Denied:
        i. System notifies the rentee of the denial.

Preconditions

- Approved + paid requests become confirmed rentals.
- Denied requests are closed and marked accordingly.

UC-5: Pickup Item

System Description: The Pickup Coordination System manages the handoff of items between renter and rentee once a booking is confirmed.

Actor: Student Rentee

- The user picking up the rented item.

Actor's Goal:

- To successfully pick up the item and activate the rental period.

Main Flow:

1. System sends a pickup reminder.
2. Users meet at the agreed pickup location.
    a. If user does not arrive
        i. System logs no show
3. Rentee inspects the item
    a. If the item does not match the listing,
        i. The rentee cancels the rental
4. Rentee confirms pickup on the Pickup Confirmation Page
    a. If the rentee does not confirm:
        i. System auto-confirms after 24 hours.
5. System marks the rental as active.

Preconditions

- The item is handed to the rentee.
- Rental enters active state.

UC-6: Return Item

System Description: The Return Management System handles the process of returning items, checking for damage, settling deposits, and closing rentals.

Actor: Student Rentee

- The user returns the rented item.

Actor's Goal:

- To complete the return process and close the rental.

Main Flow

1. System sends a return reminder.
2. Users meet at the agreed return location.
3. Renter inspects the item.
    a. If the damage is detected:
        i. System opens a dispute case.

4. Renter confirms return on the Return Page.
5. System releases the security deposit.
6. System applies late fees if applicable.
7. System closes the rental.

Preconditions

- Rental is marked as completed.
- Deposits and payments are settled.

UC-7: Request Extension

System Description: The Extension System allows rentees to request additional rental time and reroutes the request to the renter for review.

Actor: Student Rentee

- The user returns the rented item.

Actor's Goal:

- To extend the rental period before the original deadline.

Main Flow

1. User opens the Active Rental Page.
2. User selects a new desired return date.
    - If the date is unavailable:
        i. The system suggests alternatives
        ii. User selects new available dates
3. The system calculates an additional cost for the extension.
4. Renter reviews the extension request.
    - If the renter denies:
        i. The original return date remains.
5. Rentee completes extension payment.
    - If payment fails:
        i. The extension is cancelled.
6. System updates the rental end date.

Postconditions

- Rental period is extended.
- Updated charges appear in the user's payment history.

UC-8: Report a Dispute

System Description: The Dispute Management System allows users to report issues such as damage, no-shows, or misrepresentation, and provides resolution actions.

Actor: Renter or Rentee

Actor's Goal:

- To report an issue and receive a system-generated resolution.

Actor: Renter or Rentee

Main Flow

1. User opens the Report Issue Page.
2. User selects the dispute category.
3. User provides a written description and uploads evidence.
4. System notifies the other party.
5. Other party submits a response.
    a. If no response is provided:
        i. The system proceeds with available evidence.
6. System evaluates the case or escalates to an administrator (future functionality).
7. System applies the final resolution.

Postconditions

- Dispute is recorded and resolved.
- Outcome may affect deposits, ratings, or user status.

UC-9: Manage Listing Availability

System Description: The Listing Management System enables renters to update availability or remove listings while handling all pending requests.

Actor: Student Renter

Actor's Goal:

- To update, disable, or delete listing availability

Main Flow

1. User opens the My Listings Page.
2. User updates availability or marks item unavailable.
3. System saves updated availability.
4. System notifies affected rentees.
5. If the user deletes the listing:
    - System cancels all pending requests.
    - System notifies all affected rentees.

Postconditions

- Listing availability is updated.
- Pending requests are handled properly.

UC-10: AI Recommendations

System Description: The Recommendation System analyzes user behavior and listing activity to present personalized or trending item suggestions.

Actor: Student Rentee

Actor's Goal:

- To receive personalized recommendations for relevant items.

Main Flow

1. User opens the Home Page or Search Page.
2. The system analyzes user behavior and listing activity.
    - If the user has no past activity:
        i. The system shows trending items.
3. System displays recommended items.
4. The user may click a recommended item or dismiss it.
5. Dismissals are used to improve future recommendations.

Postconditions

- User receives personalized or fallback recommendations.

Functionalities

Below are the functionalities that our design would need:

User Account & Verification

F1: System allows students to register using a primary email and a university (.edu) email
F2: System verifies student status through .edu confirmation
F3: System allows secure login, logout, and password recovery

Listing Management

F4: Users can create rental listings with photos, descriptions, and prices
F5: Users can set availability calendars and rental durations
F6: Users can define pickup and return locations
F7: Users can edit, update, or remove listings
F8: Users can mark listings as unavailable

Search & Discovery

F9: Users can browse items by category
F10: System supports filtering by price, availability, location, condition, and ratings
F11: System provides an interactive campus map showing item locations
F12: System provides AI-powered recommendations based on user behavior
F13: Users can save favorites and create wish lists

Rental Request & Booking

F14: Users can request items with proposed dates
F15: System displays availability and calculates total rental cost
F16: Item owners can approve or deny rental requests
F17: System generates confirmations for both parties
F18: Users can reschedule or negotiate via in-app messaging

Payments

F19: System processes secure payments
F20: System handles security deposits
F21: System releases deposits after successful return
F22: System manages refunds when rentals are canceled or invalid

Communication

F23: Users can message each other in-app
F24: System sends notifications for all rental events (requests, approvals, reminders, late alerts)
F25: System keeps message history

Pickup & Return Logistics

F26: System sends pickup reminders
F27: Users confirm pickup in the app
F28: Users confirm return in the app
F29: System applies late fees if needed
F30: Users can request extensions

Reputation & Trust

F31: Users can rate and review each other
F32: System calculates reputation scores
F33: Users can report inappropriate behavior

Dispute Handling

F34: Users can report issues such as damage or no-shows
F35: System collects evidence from both parties
F36: System resolves disputes or escalates to admin

Future / Advanced Functionalities

F37: System integrates optional insurance for high-value items
F38: System provides fraud prevention and user verification upgrades
F39: System allows campus administrators to oversee activity (future)

Software Architectural Design

Software Quality Attributes

Below are the software quality attributes that we aim for.

1. Availability

Client Need: Users must access the system anytime, especially for pickup/return coordination.
Scenario: If the system experiences high user load or a subsystem fails, the platform stays online and responds normally.
Response Measure: 99.5% uptime or higher
Importance: Critical for time-sensitive rentals and coordinating campus meetups

2. Scalability

Scenario: If the number of listings or active users doubles during peak semester times, the system continues functioning without noticeable slowdown.
Response Measure: No more than 10% degradation in response time
Importance: The platform expands with student population and seasonal demand

3. Security

Scenario: When users log in, only authorized roles (renter, rentee, admin) access protected features. All personal information is encrypted.
Response Measure: Zero unauthorized access; 100% encryption
Importance: Protects student identity and financial transactions

4. Usability

Scenario: A new user should be able to find and request an item in under 3 steps without assistance.
Response Measure: ≥90% task completion during usability tests
Importance: Students need a simple, intuitive interface

5. Maintainability

Scenario: Developers can deploy a small update (bug fix, UI change) without taking the system offline.
Response Measure: Updates deploy in < 30 minutes with zero downtime
Importance: Ensures system stability throughout the academic year

6. Data Accuracy

Scenario: Availability calendars and rental status must always reflect the correct user actions without conflict or delay.
Response Measure: ≥95% accuracy in operations
Importance: Prevents double-booking and rental disputes

Priority Ranking with Justification:

1. **Security** (Highest Priority)
    - Justification: Without strong security, the platform cannot protect student data, financial information, or prevent fraud. A security breach would destroy user trust and could expose the university to liability. This is foundational—all other attributes depend on users feeling safe using the platform.
2. **Availability**
    - Justification: If the system is down during critical times (pickup coordination, exam week rentals), students cannot complete transactions. High availability ensures students can rely on the platform for time-sensitive needs, which is essential for a campus service.
3. **Data Accuracy**
    - Justification: Incorrect availability or double-booking undermines the entire rental model and leads to disputes between students. Accurate data is critical for maintaining trust and preventing conflicts that could harm the community.
4. **Usability**
    - Justification: Students will abandon a confusing platform in favor of alternatives. Good usability drives adoption and reduces support overhead, making the platform sustainable.
5. **Scalability**
    - Justification: While important for growth, initial user base is limited to one campus. Scalability becomes more critical as the platform expands, but in early phases, other attributes take precedence.
6. **Maintainability**
    - Justification: Important for long-term sustainability but less immediately critical than security or availability. A maintainable system reduces costs over time and enables feature evolution, but it can be improved iteratively.

Architectural Design Analysis

Overview

After analyzing our quality attribute priorities and system requirements, we have chosen a hybrid architectural approach that combines the Three-Layer (Layered) Architecture as the primary structure, with selective Microservices for critical subsystems especially for the data layer and the MVC Pattern within the presentation layer. This combination directly supports our top quality attributes: Security, Availability, and Data Accuracy.

Primary Architecture: Three-Layer Pattern

Selection Rationale

The Three-Layer Architecture separates our application into three distinct layers with clear responsibilities. This pattern provides maintainability through independent layer updates, security through centralized business logic, and data accuracy through a single source of truth for rental data.

Layer Responsibilities

Presentation Layer (User Interface)

- Displays all user-facing pages including registration, search, listings, and rental dashboards
- Renders interactive campus maps and real-time notifications
- Handles form inputs and basic client-side validation
- Formats data for display such as prices, dates, and availability calendars
- Communicates with Domain Layer via REST API calls

Domain Layer (Business Logic)

- Enforces all business rules and rental policies
- Manages the complete rental workflow from request through approval, payment, pickup, return, and completion
- Calculates rental costs, security deposits, and late fees
- Validates rental requests for date conflicts, user eligibility, and item availability
- Coordinates payment processing and notification delivery
- Updates user reputation scores based on ratings and completed transactions
- Handles dispute resolution logic

Data Source Layer

- Stores and retrieves all application data including users, listings, rentals, reviews, messages, and disputes
- Encrypts sensitive information such as passwords and payment details
- Manages database connections, transactions, and query optimization
- Integrates with external services including email verification, SMS, and mapping APIs
- Provides data access through repository interfaces that abstract database operations
- Includes multiple specialized databases: Main PostgreSQL database for core data, separate databases for Payment, Notification, Search, and Authentication microservices

Layer Communication

Layers communicate only with adjacent layers. The Presentation Layer calls the Domain Layer via REST API. The Domain Layer accesses the Data Source Layer through repository interfaces and microservice APIs. The Presentation Layer never directly accesses databases, and the Data Source Layer never renders UI components.

Secondary Architecture: Selective Microservices

Selection Rationale

While the core platform uses the three-layer architecture, certain critical subsystems operate as independent microservices within the Data Source Layer. These microservices have their own databases and can scale independently. We selected microservices for components requiring high security isolation, independent scaling capabilities, different failure tolerance levels, or third-party integrations.

Microservice Components

Payment Processing Service

- Handles all financial transactions including rental payments, security deposits, refunds, and late fees
- Separated for PCI compliance, security isolation, and to prevent payment failures from affecting core platform operations
- Communicates with Domain Layer via REST API calls

Notification Service

- Manages all user notifications including rental requests, approvals, denials, pickup reminders, and return warnings
- Sends notifications through multiple channels including email, SMS, and push notifications
- Separated to handle high notification volumes during peak times without impacting rental operations
- Operates asynchronously by subscribing to events published by the Domain Layer

Search and Recommendation Service

- Indexes all rental listings and processes search queries with filters for price, location, category, and availability
- Generates personalized AI-powered recommendations based on user behavior and preferences
- Uses Elasticsearch for optimized full-text search and geolocation queries
- Separated to scale independently during high search traffic periods

Authentication and Authorization Service

- Manages user registration, login, session management, and role-based access control
- Verifies student status through university email confirmation
- Validates authentication tokens for all API requests
- Separated to isolate critical security concerns and enable potential reuse across other campus systems

Why Not Pure Microservices

We rejected a full microservices architecture because the rental workflow is inherently sequential and requires centralized coordination. Managing numerous independent services would introduce excessive complexity, network overhead, and maintenance burden for our team size and project timeline. The rental process from request through approval to payment and completion benefits from coordinated state management in a single Domain Layer.

Supporting Pattern: MVC Within Presentation Layer

Selection Rationale

The Model-View-Controller pattern organizes code within the React-based Presentation Layer to separate concerns and maintain clean architecture.

MVC Components

Model (Application State)

- Redux store managing all client-side application state
- Caches data from backend API to reduce unnecessary network calls
- Stores current user information, search results, active rentals, messages, and UI state
- Triggers automatic re-rendering when state changes

View (UI Components)

- React components displaying data to users
- Includes pages for search, listing details, rental dashboards, and messaging
- Purely presentational with no business logic
- Receives data from Model and emits user interaction events to Controller

Controller (Event Handlers)

- Handles all user interactions including clicks, form submissions, and navigation
- Makes API calls to the Domain Layer backend
- Validates user input on the client side
- Updates Model with API responses
- Manages loading states and error handling

Data Flow

Data flows unidirectionally from View to Controller to Model and back to View. When a user interacts with the UI, the View component emits an event that the Controller captures. The Controller makes necessary API calls to the Domain Layer, receives responses, and updates the Model. Model changes automatically trigger View re-renders to reflect the updated state.