REQUIREMENTS DOCUMENT:
APARTMENT RENT BIDDING SYSTEM

PROJECT INFORMATION

Project Name: Apartment Rent Bidding System
Submission Date: October 25, 2022
RD Version: 1.0
Client: Damen Tomassi

PROJECT TEAM

Team Member	Role	Document Responsibilities
Gregory Zacharko	Product Owner	Data Management; Change Control; Common Terms and Definitions Within Project Scope and Definition Section; Non-Functional Requirements; Proofread RD
Lucas Adams	Scrum Master	Constraints; Burn-Up Chart
Alizsa Johnson	Developer	Project Scope and Definition
Emma Dougherty	Developer	System Requirements and Functional Requirements; Non-Functional Requirements
Sara Torres	Developer	System Environment: Use Case Diagram; External Interface Requirements

PROJECT SCOPE AND DEFINITION

The Apartment Rent Bidding System is a Web application that allows apartment administrators to auction off apartment units to potential renters.

The administrators create apartment buildings, populate those buildings with apartment units, and set the minimum going price and bid rate. The apartment administrator gives the bidders the apartment unit ID. Then the bidders start placing bids and continue bidding until the auction ends. At the end of the auction, a winner is selected and both the winning bidder and the apartment administrator are notified.

The scope of the Apartment Rent Bidding System is to create a Web application where apartment administrators can hold auctions for apartment units and invite bidders to bid on the rent.

For Administrators

- Allow administrators to sign up for a special user account that allows them to create apartment buildings and display available units.
    - Once an administrator creates an apartment unit, it is assigned a unique identifier that is visible to the administrator. This identifier allows approved bidders to access the auction.
- Allow administrators to set the minimum bid and minimum bidding increment.
- Allow administrators to add apartment details such as lease length and amenities.

For Bidders

- Allow bidders to sign up for an account that lets them participate in auctions.
- Allow bidders to bid on apartment units.
- Allow bidders to use an auto bidding feature, where they set the max amount they'd like to spend and the bidding system will bid on the users behalf until no other competing bids are placed or until the bidders max price is reached.
- Allow bidders to be notified when they've been outbid and when they've won the auction.

Deliverables

This project will yield a fully functional Web application that allows apartment administrators to auction off apartment units and allows bidders to access and participate in the auctions.

Milestones

- Sprint 0:
    - Decide what kind of pages need to be present on the web application and create the GUI wireframes in Figma.
    - Research the features of Bubble.io.
- Sprint 1:
    - Recreate the Figma diagrams in Bubble.io and begin adding functionality to the web pages.
    - Set up the database for the apartment listing details so that it stores admin input and displays it on the web page.
    - Verify that administrators can create listings.
- Sprint 2:
    - Set up additional databases that will hold the account information for the bidders and administrators and verify that both user types are able to sign up and log in.
    - Create the bidding feature that will allow bidders to participate in the auction.
- Sprint 3:
    - Create the auto-bidding feature and decide how both bidders and the administrator will be notified of significant events related to the auction.
    - Look into using a plugin to send out email updates to the auction participants.
- Sprint 4:
    - Begin bug testing the web application. All of the key features will be implemented so the next step is to verify that the application is working as intended.
    - Begin implementing stretch goals.
- Sprint 5:
    - Test any added features and confirm that they are working properly.

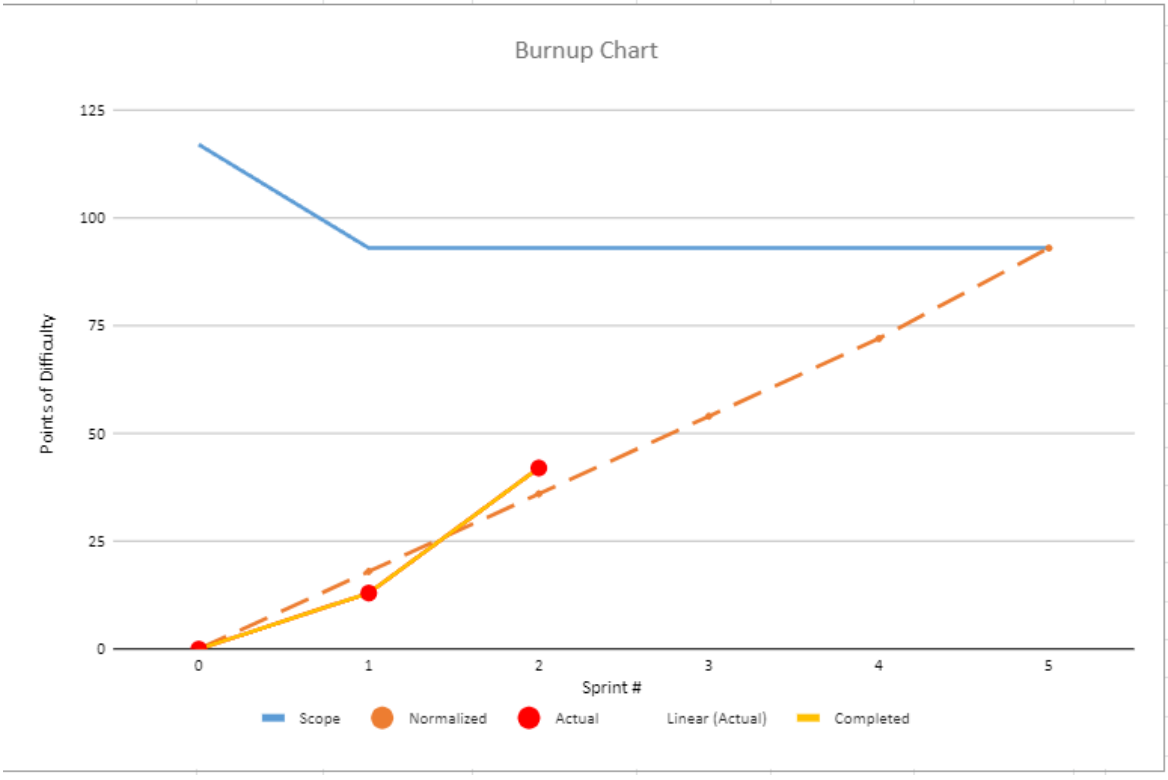
COMMON TERMS AND DEFINITIONS:

1. SPONSOR: The owner/stakeholder of the product.
2. PROFESSOR: Third-party overseer of the project and the team developing it.
3. ADMINISTRATOR: A person that creates a listing for an apartment unit using the product.
4. BIDDER: A person that uses the product to bid on an open and available apartment unit.
5. APARTMENT UNIT ID: A randomized, unique sequence of letters and numbers that bidders use to access the specific apartment listing they want to bid on.

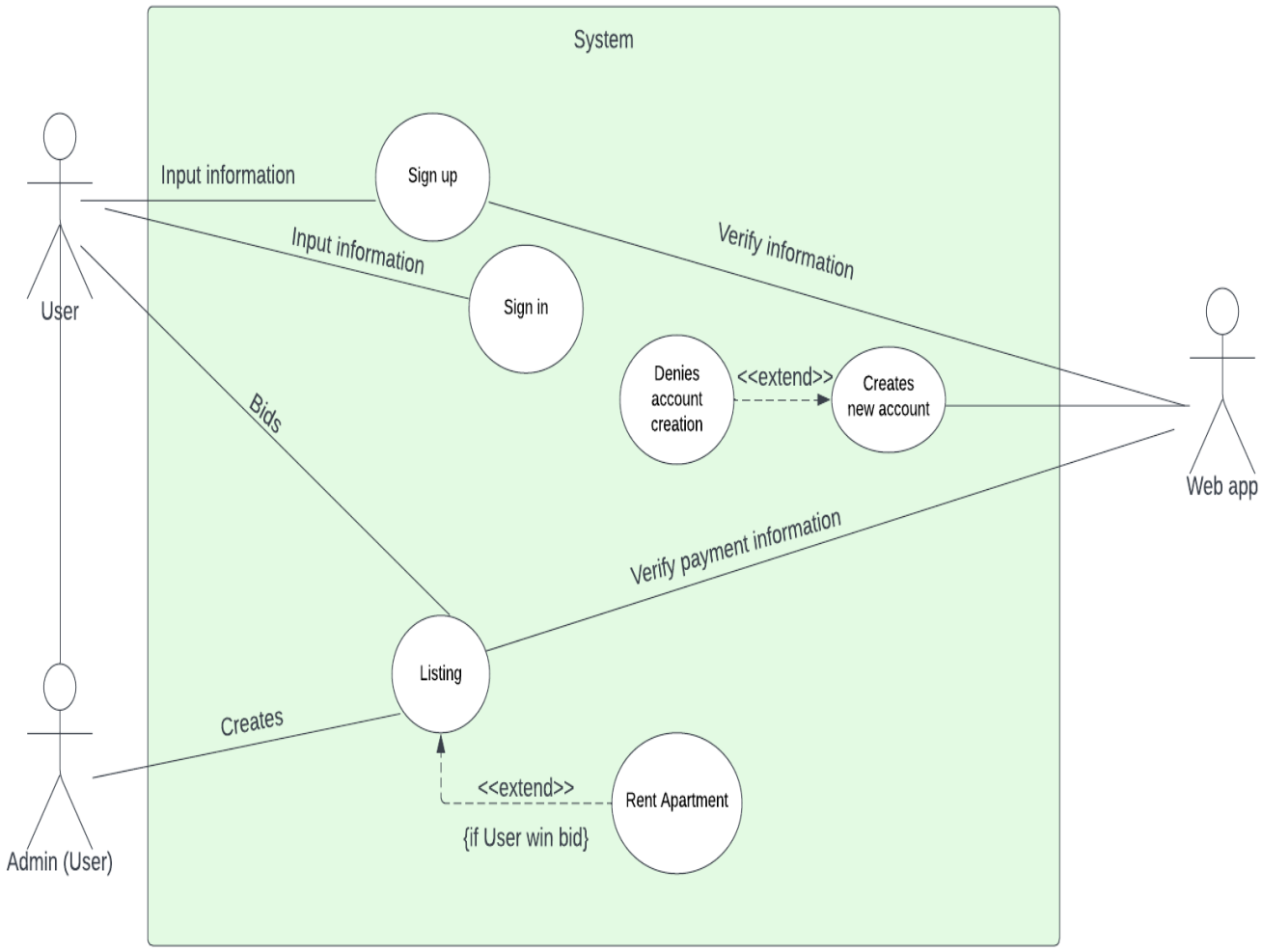
6. BUBBLE.IO: A software engineering platform that allows developers to quickly build high-quality, fully functioning, powerful desktop/Web and mobile applications without having to type out thousands of lines of code.
7. ELEMENT: A component on the website/Web application that the user can interact with (e.g. buttons, links, text input boxes) or that displays information to the user (e.g. dynamic text displays).
8. ELEMENT PALETTE: The place where a developer can pick an element to drag-and-drop onto the page. It is where all possible elements are listed.
9. WORKFLOW: The different user interactions that can occur between the administrators and bidders, and the website/Web application. It is the sequence/flow of events and interactions when an administrator and/or bidder uses the product.
10. FIGMA: A software platform that allows developers to create a wireframe for their project/product's GUI.

BURN-UP CHART

Sprint	0	1	2	3	4	5
Scope	117	93	93	93	93	93
Estimated	0	18	36	54	72	93
Completed	0	13	29	#N/A	#N/A	#N/A
Cumulative	0	13	42	#N/A	#N/A	#N/A



SYSTEM ENVIRONMENT: USE CASE DIAGRAM



SYSTEM REQUIREMENTS AND FUNCTIONAL REQUIREMENTS

The overall functionality of the Apartment Bidding System includes a graphical user interface that allows an apartment bidder to bid on a specific apartment and an administrator that can input listings of apartments for bidding. There are several pages within the application that detail different functionalities depending on the user of the system.

Functional Requirements	Description
Bubble.io Database	Adds data to certain fields based on buttons pressed and information entered by the user. This includes fields from administrators and bidders.
Preview Listing Page (Admin Listing Page)	Admins can preview a listing that they have created with the information they inputted displayed on the screen.
Virtual Tour of the Apartment (User Listing Page)	The user may take a virtual tour of the apartment with 360 degree views of the apartment with draggable images.

EXTERNAL INTERFACE REQUIREMENTS

Our user interfaces' style guides consist of:

- Typography:
  - Font: Raleway, App Font(Open Sans), Montserrat.
  - Font sizes: 14px, 16px, 24px, 50px.
- Iconography: Icons are a very helpful and simplistic way to improve the accessibility of a product. They can be paired with typography to clarify what an element does or where it will send a user. We are using icons of buildings, calendars, etc.
- Layouts and Grids: Layouts and grids are frameworks to organize the visual elements of our digital product. Our layout consists of a standard website size for browsers.
- Color Palette: Color plays a vital role in how a user feels when using an app or other interface, which is why it is important that whatever color palette we choose stays consistent throughout the end-to-end experience. We have chosen the colors olive green, light and dark greens, and white, to give a comfortable professional atmosphere to our website.
- Components: The components we choose and how we decide to display them plays a significant role in the user experience. We have a consistent pattern for our website design, having round borders and having slight differences between the website experience for a user and for an admin. We also implemented easy-to-see buttons and pop up pages to make the user experience as easy, efficient, and clean as possible.

A user interface, also sometimes called a human-computer interface, comprises both hardware and software components. It handles the interaction between the user and the system. Our platform is basically between a type of marketplace, utility platform, and interaction network.

- Utility Platforms: Utility Platforms attract users by providing a useful, typically free, service. Our admins will be free to create an account to put their apartment listings up, and our users will be free to create an account to start bidding on those apartment listings.
- Interaction Networks: The common element is that this type of platform facilitates interactions between specific participants (people and/or businesses). In our website, this will be minimal but important since each user gets provided with a code for a listing by the admin, and later, when confirmed, they can then ask the admin any questions they have.
- Marketplace: Our website is mainly concentrating on a marketplace platform. Marketplaces enable transactions between demand-side participants (buyers) and supply-side participants (sellers).

Our database system is facilitated by Bubble.io. We are using a Listings database, User database, and Admin database to store apartment listing information, user account information, and admin account information. Any device that has Internet and access to a browser will be able to access our website.

SYSTEM, DESIGN, AND DEVELOPMENT CONSTRAINTS

System, Design, Development	Constraint
Standards	The only software required is Bubble.io.
Schedule	The deadline for the product to be fully complete is by the end of Sprint 5 (which should be 12/1). The deadline for the product to be fully functional, excluding minor issues, is by the end of Sprint 4 (which should be 11/17).
Budget	The only budgetary concern is that a Bubble.io subscription may be required, however currently this seems unlikely.
Preferred Software Programming	Because we are using Bubble.io, the

Language	programming language is Javascript.
Required Development Tools	The only development tool required is Bubble.io.
Handling Security Requirements	The only security issue would be user information leaking, however Bubble.io has secure fields, so everything would be covered there.
Potential Risks	The Team not having enough time to build some features of the product. These time limitations may affect the development of potential features and functionality originally planned to be in the product. Since we are using Bubble.io, the only way to integrate 3rd party systems into the product is through Bubble.io plugins for those 3rd party services. Thus, if a 3rd party application does not have a plugin for Bubble.io, then it is not possible to integrate it into the product. This limits the customization and ability for the Team to be able to build certain functionality into the product.
Policy and Regulation	There are currently no plans for website policies, however if the system were to move on past the scope of the Sprints, a system can be put in place to verify user information such as their e-mail and phone number.

NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirement	Description
Performance Requirements	The response time is based on the functionality of Bubble.io and the user's internet connection strength and speed. The response time may vary depending

	on the amount of workflows present and the actions of the users as well. For example, when a button is pressed, the response time for that button must be under 10 seconds. The response time for changes made to the user and listing data in the database must be under a minute.
Capacity	The system's storage requirements include an unlimited amount of storage as stated on Bubble.io's webpage. The only storage that is limited is for uploading files, which is limited to 50 MB per file.
Resources	The system will implement a database built into Bubble.io for storing data. This database will be connected to certain functionalities within the system, including when a user enters any important information. Communications between the user and the website are performed over the Internet.
Centralized or Distributed Processing	The Bubble.io database is a centralized database. It is only accessible on Bubble.io. Processing on the data in the database is done within Bubble.io.
Security Requirements	Bubble.io is an HTTPS secure website, and thus all traffic to a Bubble.io-made site is also HTTPS encrypted. Additionally, Bubble.io automatically ensures worldwide accessibility and protection against Distributed Denial of Service (DDoS) attacks through Cloudflare and Amazon Web Services (AWS). Passwords data for Bubble.io sites are salted and encrypted by Bubble.io as well, so users of our site can confidently sign up without worry about security breaches or data leaks.
Compatibility	The minimum hardware requirements are simply a device, like a computer, with access to the Internet and a web browser such as Google Chrome.

Scalability	Bubble.io automatically handles web traffic and will scale up accordingly if the site sees any spikes in activity.
Availability	The site is accessible with any Internet connected device that has a web browser, such as Google Chrome.
Inputs	Our site utilizes pop-up windows for users and admins to enter information.
Add Listing Button (Admin Dashboard)	An administrator can create listings using the “Add Listing” button. This button creates a pop-up where the admin can input the listing information.
Admin Dashboard Pop-Up (Admin Dashboard)	Information entered by the admin on this pop-up includes pictures of the apartment, the name of the apartment, the address of the apartment, how many bedrooms the apartment has, how many bathrooms the apartment has, the square footage of the apartment, and a starting bid for the apartment.
See Bidders Button (Admin Listing Page)	The “See Bidders” button allows admins to view the bidders along with the amount they bid and their contact information.
Sign Up Button (Home Page)	Information needed by the user to sign up includes an email address, password, phone number, first and last name (full name), and the account type needed (administrator or bidder). The user will enter this information by pressing the “Sign Up” button.
Account Type Radio Button (Home Page)	Account type (admin or bidder) will need to be chosen by the user. Depending on the choice of this button, a user will be brought to their appropriate dashboard.
Sign Up Pop-Up (Home Page)	The user will enter an email address, password, phone number, and first and

	last name (full name) into the fields on the pop-up screen.
Create Account Button (Home Page)	The “Create Account” button on the sign up pop-up will then enter all of the information given into the database.
Sign In Button (Home Page)	After signing up, the user must log into their account using the “Sign In” button. The user must enter their email and password into the fields. Functionality for the sign-in page includes password verification and account type recognition. The user will then be brought to their designated dashboard page: the user dashboard page, or the administrator dashboard page.
Bid Button (User Listing Page)	The “Bid” button must then display a pop-up where the user can enter their bid as well as their email for account verification.
Contact Admin Button (User Listing Page)	The user may contact an administrator with any questions or concerns about the apartment they may have, using the “Contact Admin” button. A pop-up may be displayed to do so.
Environmental	Our site is expected to only perform within the environment of any sort of web browser, like Google Chrome.

DATA MANAGEMENT

Data Requirement	System Environment	Data
Website/Web Application	Bubble.io Hosts and Deploys Website/Web Application	Drag-and-Drop Elements From the Element Palette: - Buttons - Links

		- Images - Videos - Text Input Boxes - Dynamic Text Displays Imported GUI Wireframe From Figma
Bubble.io Elements Tree	Bubble.io	All The Elements Currently On That Page; Relationships Between Elements and Its Contents
Bubble.io Workflow	Bubble.io	User Interactions: - Clicking Button Elements - Typing Text Into A Textbox - Signing Up For An Account - Logging In and Out of Account - Entering Listing Page via Code - Submitting Manual Bids - User AutoBidding on Listing - Admin Creating Listing - Admin Adding Text Descriptions, Images, and Video to Apartment Listing Page - Admin Inputting Apartment Information Into Textboxes - Admin Inputting Apartment Contract Information Into Textboxes - Modifying User/Admin Account

		Personal Information - Contacting Admin and Winning Bidder When Auction Is Over - Contacting An Outbid Bidder
GUI Wireframe	Figma	GUI Structure Blueprint
Database	Bubble.io Hosts Database	User Account Data/Fields: - Account Type - First Name - Last Name - E-Mail - Phone Number - Password - AutoBid On/Off - AutoBid Increment - AutoBid Maximum Limit Admin Account Data/Fields: - Account Type - First Name - Last Name - E-Mail - Phone Number - Password Apartment Listing Fields: - Apartment Listing ID/Code - Date and Time of Auction Start - Length of Auction - Address - Apartment Number - Minimum Price - Lease Length - Number of Amenities/Bedrooms /Bathrooms - Apartment Description/Text Inputted By Admin

		- Images Uploaded By Admin - Videos Uploaded By Admin - Current Highest Bid - Minimum Bid Increment Allowed By Admin
Text	Bubble.io	Apartment Description and Information (Text)
Images	Bubble.io	JPG; PNG
Video	Bubble.io	MP4; YouTube

CHANGE CONTROL

Changes to the project's scope are born from the Sponsor and from the Project Team. The Sponsor may ask the Team to add additional functionality or features to the product, such as implementing Google Maps into the product to allow administrators to show a map of the precise location of the apartment on the apartment listing page for users to be able to click and interact with and to compare how far away they are from the apartment's location. Likewise, the Sponsor could also ask for extraneous features to be removed for the sake of the project's budget, product size, the timeline/time frame of the project, and product efficiency. These types of requests can increase or decrease the scope of the project astronomically. The Project Team itself can also conjure up new ideas for functionality or scrap features that are not necessary to the core functions of the product. The Team can potentially make these kinds of decisions without the Sponsor's permission, however, the Team makes the decision together with input from everyone within the Project Team. But if the Sponsor demands that a certain feature be included or not, or for certain project parameters to be maintained, such as meeting a certain efficiency threshold, then the Team must comply with those requirements. Any changes made to the scope of the project are controlled and tracked by both the Burn-Up Chart and the Trello Board/Product Backlog. If either the Sponsor or the Team feel that something is off with the potential direction or scope of the project at that point in time, then possible changes would need to be discussed and then approved by both parties. Ultimately though, based on how the Product Backlog and the Burn-Up Chart look, and how/if the Team is meeting the Sponsor's requirements and expectations for the project, both the Sponsor and the Project Team will need to agree, approve, and be happy with the direction and scope of the project at that time, and with how the health of the project is projected to be in the future.