DELIVERABLE 1: DOCUMENTATION

SFWRENG/MECHTRON 3K04 - Software Development

L04 - Group 8

Fall 2025

Shiv Patel (400530101)

Nihal Inel (400448431)

Marwan Sayed (400509757)

Muhammad Haseeb Aslam (400449291)

Mathura Nambyarooran (400373319)

Elijah James (400510634)

Submitted On: October, 27 2025

Table of Contents

List of Tables

List of Figures

2 Part 1

2.1 Introduction

The pacemaker system is a medical device designed to simulate how an implantable cardiac pacemaker helps regulate a person's heartbeat by sending electrical pulses to the heart's chambers. It consists of two main components: the Pacemaker and the Device Controller and Monitor (DCM). The Pacemaker is based on and built using Simulink, as it controls the pacing behaviour in different modes, while the DCM acts as the user interface (UI) that lets users log in, register, and adjust various pacing parameters.

The overall objective of this deliverable was to create a simulated version of both components and document the process as well. This process involves implementing the Pacemaker in AOO, VOO, AAI, and VVI modes, and developing the DCM's front-end to display these modes and allow for their key programmable parameters to be saved. The documentation is designed to clearly explain the system's requirements, design choices, testing and validation reports, and discuss any future changes or plans.

The scope of this part is limited to the initial design and development of the Pacemaker and DCM systems. Communication between the two systems is not implemented yet, as that will be the focus of the next deliverable. The key focus in this deliverable was to understand the system requirements, apply software design principles, and establish a solid base to expand upon.

2.2 Requirements

2.2.1 Pacemaker - Overall System Requirements

The pacemaker is designed to monitor a patient's heart rate, ensuring it stays within a healthy range. When the device detects bradycardia, a type of abnormal heart rhythm, it delivers pacing therapy to restore and maintain a normal rhythm [1]. It can operate in both single and dual-chamber pacing modes, and it adjusts automatically based on the patient's movement, which is determined using an accelerometer that senses changes in body activity.

Along with the real-time monitoring, the system also keeps a record of historical data, such as sensor readings and atrial and ventricular rate histograms. This data plays a key role, as it can later be reviewed for further analysis through its bradycardia analysis feature. The pacemaker can be programmed wirelessly through two-way telemetry with the DCM, making it easy to update settings or retrieve information without any complex procedures [1].

Its analysis features provide detailed pacing measurements like lead impedance, pacing thresholds, P and R wave readings, battery status, temporary pacing support, and motion sensor trends [1]. Additionally, the pacemaker and the DCM both include built-in diagnostic tools such as telemetry markers, intracardiac electrograms (EGMs), repeated wave measurements, and battery performance tests to validate their operation.

Lastly, the pacemaker is designed to meet electrical safety standards and operate safely alongside common hospital equipment like ECG monitors, fluoroscopes, anesthesia machines, patient water blankets, electrocautery devices, and pulse oximeters [1].

2.2.2 Pacemaker - Design Requirements

The pacemaker design software must implement the bradycardia pacing modes AOO, VOO, AAI, and VVI based on a given set of user-defined parameters, allowing for patient-tailored bradycardia therapy. The stateflow for these pacemaker modes uses the following relevant programmable parameters.

Parameter	AOO	AAI	VOO	VVI
Lower Rate Limit	X	X	X	X
Upper Rate Limit	X	X	X	X
Atrial Amplitude	X	X		
Ventricular Amplitude			X	X
Atrial Pulse Width	X	X		
Ventricular Pulse Width			X	X
Atrial Sensitivity		X		
Ventricular Sensitivity				X
VRP				X
ARP		X		
PVARP		X		
Hysteresis		X		X
Rate Smoothing		X		X

Table 1: Programmable Parameters for Bradycardia Therapy Modes

Mode	Chamber Paced	Chamber Sensed	Response To Sensing
AOO	A: Atrium	O: None	O: None
VOO	V: Ventricle	O: None	O: None
AAI	A: Atrium	A: Atrium	I: Inhibited
VVI	V: Ventricle	V: Ventricle	I: Inhibited

Table 2: Bradycardia Operating Modes

The detailed operation of each mode is described in the sections (2.2.2.1 - 2.2.2.4) below.

2.2.2.1 Mode: AOO

In AOO mode, the pacemaker delivers atrial pacing pulses at a fixed rate determined by the Lower Rate Limit (LRL), independent of any intrinsic heart activity. Periodic pacing spikes should appear in the atrial signal at consistent intervals corresponding to the programmed rate. This mode does not sense or respond to heart activity.

2.2.2.2 Mode: VOO

In VOO mode, the pacemaker delivers ventricular pacing pulses at a fixed rate determined by the LRL, independent of any intrinsic heart activity. Periodic pacing spikes should appear in the ventricular signal at consistent intervals corresponding to the programmed rate. This mode does not sense or respond to heart activity.

2.2.2.3 Mode: AAI

In AAI mode, the pacemaker monitors intrinsic atrial activity and only delivers pacing pulses when no natural atrial beat is detected within the interval defined by the LRL. This mode senses atrial activity and responds by inhibiting pacing when intrinsic atrial beats occur.

2.2.2.4 Mode: VVI

In VVI mode, the pacemaker monitors intrinsic ventricular activity and only delivers pacing pulses when no natural ventricular beat is detected within the interval defined by the LRL. This mode senses ventricular activity and responds by inhibiting pacing when intrinsic ventricular beats occur.

2.2.3 DCM - Design Requirements

The DCM is the user interface for configuring and monitoring the pacemaker system. For Deliverable 1, the DCM implements the front-end of the application. It includes user authentication, parameter management, and placeholder status indicators. The DCM is developed in Python using the Tkinter GUI framework.

2.2.3.1 User Authentication Requirements

The DCM shall have a secure user authentication procedure, including the following capabilities:

- User Registration: The system shall allow new users to register with a unique username and password combination. A maximum of 10 users may be stored in the local database.
- User Login: Registered users should be able to log in using their saved usernames and passwords.
- Password Security: User passwords shall be securely hashed using bcrypt encryption before storage to protect against unauthorized access.
- User Data Persistence: Each user's programmable parameters shall be saved independently and persist across sessions.

2.2.3.2 User Interface Requirements (PACEMAKER 3.2.2)

In accordance with section 3.2.2 of the PACEMAKER System Specification, the DCM user interface should be able to perform the following:

- **Window Management:** The user interface shall utilize and manage windows for displaying text and graphics.
- **User Input Processing:** The user interface shall process user positioning and input buttons, including mode selection, parameter entry, connection controls, and action buttons.
- **Parameter Display:** The user interface shall display all programmable parameters relevant to the selected pacing mode for review and modification, with clearly labelled fields and valid range indicators.
- **Communication Indication:** The user interface shall visually indicate when the DCM and the pacemaker device are communicating through a connection status indicator (green = connected, red = disconnected).
- **Out-of-Range Indication:** The user interface shall visually indicate when telemetry is lost due to the device being out of range through an orange telemetry indicator and a corresponding warning message.
- **Noise Indication:** The user interface shall visually indicate when telemetry is lost due to electromagnetic interference or noise through an orange telemetry indicator and an appropriate warning message.
- **Device Change Indication:** The user interface shall visually indicate when a different PACEMAKER device is approached than was previously interrogated through a warning label and message box notification.

2.2.3.3 Pacing Mode Requirements

The DCM shall provide user interfaces for all four bradycardia pacing modes implemented in the pacemaker as described in sections 2.2.2.1 - 2.2.2.4. The system shall show the parameters of the selected mode and hide irrelevant parameters for a user-friendly interface and to prevent input errors.

2.2.3.4 Programmable Parameter Requirements

The DCM shall display input fields and validation for the programmable parameters as specified in Appendix A of the PACEMAKER System Specification [1].

Parameter	Required For Modes	Range	Units	Increment
Lower Rate Limit	AOO, VOO, AAI, VVI	30-175	ppm	5 ppm (30-50), 1 ppm (50-90), 5 ppm (90-175)
Upper Rate Limit	AOO, VOO, AAI, VVI	50-175	ppm	5 ppm
Atrial Amplitude	AOO, AAI	0.0-7.0	V	0.1 V (0.5-3.2), 0.5 V (3.5-7.0)
Atrial Pulse Width	AOO, AAI	0.05-1.9	ms	0.05 ms, 0.1 ms
Ventricular Amplitude	VOO, VVI	0.0-7.0	V	0.1 V (0.5-3.2), 0.5 V (3.5-7.0)
Ventricular Pulse Width	VOO, VVI	0.05-1.9	ms	0.05 ms, 0.1 ms
VRP (Ventricular Refractory Period)	VVI	150-500	ms	10 ms
ARP (Atrial Refractory Period)	AAI	150-500	ms	10 ms

Table 3: Programmable Pacemaker Parameters and Settings by Mode

Additional parameters supported for future implementation:

- PVARP (Post-Ventricular Atrial Refractory Period): 150-500 ms
- Hysteresis Rate Limit: Off or same as LRL range
- Rate Smoothing: Off, 3%, 6%, 9%, 12%, 15%, 18%, 21%, 25%
- Atrial Sensitivity: 0.25-10.0 mV

- Ventricular Sensitivity: 0.25-10.0 mV

2.2.3.5 Parameter Validation Requirements

The following are enforced validation rules:

- Range Validation: All parameter values shall be validated against their specified minimum and maximum ranges before saving or programming to the device.
- Type Validation: All parameter inputs shall be validated as numeric values of the appropriate type (integer or floating-point).
- Error Handling: The system shall display clear error messages indicating which parameters are out of range or invalid when validation fails.
- Save Prevention: The system shall prevent saving or programming of invalid parameter values, requiring user correction before proceeding.

2.2.3.6 Device Control Requirements

The DCM shall provide the following device control functions: Connect to Device (establish simulated connection), Disconnect (terminate connection), Interrogate Device (retrieve parameters), Program Parameters (send validated parameters to device, requires connection), and Reset to Nominal (restore default values for current mode).

2.2.3.7 Data Persistence Requirements

User parameters shall be stored in JSON format as data/params_<username>.json, including the selected pacing mode. Parameters are loaded at login or initialized to nominal defaults if no saved data exists.

2.2.3.8 Status and Monitoring Requirements

The DCM shall display: Connection Status (Connected/Disconnected with green/red indicator), Telemetry Status (OK/Lost with green/orange/gray indicator), Device ID (unique identifier), and a temporary Device Change Warning when a different device is detected.

2.2.3.9 Scope Limitations for Deliverable 1

The following are not implemented in Deliverable 1: serial communication with hardware (connections and programming are simulated), real-time EGM display (data structures only), actual device parameter transmission, and advanced diagnostics (battery status, lead impedance, threshold tests). Full hardware integration is planned for Deliverable 2.

2.3 Design

2.3.1 System Architecture (Simulink)

The high-level design is divided into 3 main components: the Input subsystem (programmable parameters and hardware inputs), the Stateflow model, and the output subsystem (hardware outputs). The inputs define programmable values for timing and amplitude parameters. The stateflow model controls the pacing output depending on the current mode (AOO, AAI, VOO, VVI). The figure below demonstrates how Simulink allows only user-programmable variables to remain accessible through the interface while embedding the rest of the code and state flow within the system.

Hardware Hiding occurs when Simulink maps between the hardware pins and the input/output variables for the four pacing modes (AOO, VOO, AAI, and VVI) on the FRDM-K64F board. The stateflow model can be seen as a block box, receiving inputs, processing and sending outputs.

2.3.2 Programmable Parameters

Parameter	Default Value	Units	Description
Mode	3	N/A	Sets default mode
Atrial Amplitude	3.5	V	Amplitude of the pulse delivered when pacing the atrium.
Ventricular Amplitude	3.5	V	Amplitude of the pulse delivered when pacing the ventricle.
Atrial Pulse Width	10	ms	Duration of the pulse delivered when pacing the atrium.
Ventricular Pulse Width	10	ms	Duration of the pulse delivered when pacing the ventricle.
Lower Rate Limit	60	BPM	The lowest heart rate the pacemaker needs to maintain. It is also used as the fixed heart rate during non-adaptive modes.
ARP	250	ms	Time interval after pacing or sensing an atrial pulse during which the pacemaker neglects any additional signals from the atrium.
VRP	320	ms	Time interval after pacing or sensing a ventricular pulse when the pacemaker disregards any further ventricular signals.
ATR_CMP_REF_PWM	90	mV	PWM is used to charge a capacitor that maintains a steady voltage for comparison, allowing the system to differentiate real heart signals from noise.
VENT_CMP_REF_PWM	90	mV	PWM is used to charge a capacitor that maintains a steady voltage for comparison, allowing the system to differentiate real heart signals from noise.

Table 4: Operating Values for Programmable Parameters

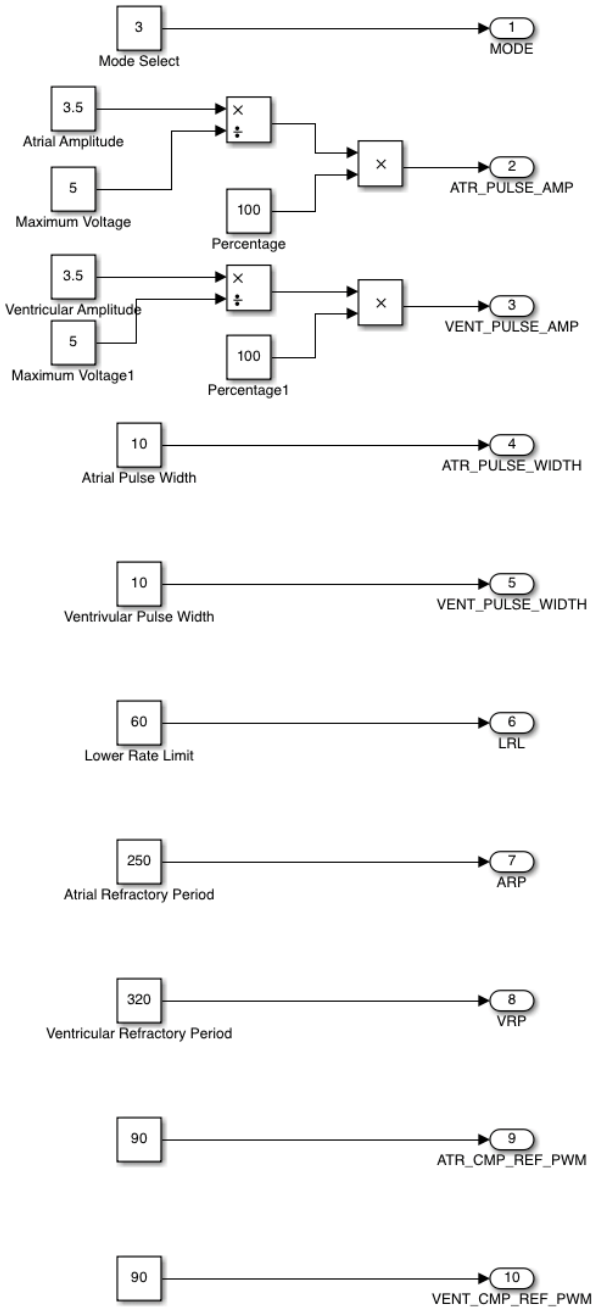


Figure 1: Hardware Programmable Parameters Subsystem

2.3.3 Hardware Inputs & Outputs

Pin	Signal Name	Direction	Function
D0	ATR_CMP_DETECT	Input	Input for Atrial Sense Comparator
D1	VENT_CMP_DETECT	Input	Input for Ventricular Sense Comparator
D2	PACE_CHARGE_CTRL	Output	Control pacing capacitor charging
D4	Z_ATR_CTRL	Output	Connects impedance circuit to Atrium's ring electrode
D5	PACING_REF_PWM	Output	PWM Voltage control for Amplitude
D7	Z_VENT_CTRL	Output	Connects impedance circuit to Ventricle's ring electrode
D8	ATR_PACE_CTRL	Output	Controls Atrial Pacing electrode
D9	VENT_PACE_CTRL	Output	Controls Ventricular Pacing electrode
D10	PACE_GND_CTRL	Output	Connects Ground Path for Pacing
D11	ATR_GND_CTRL	Output	Connects path for discharge
D12	VENT_GND_CTRL	Output	Connects path for discharge
D13	FRONTEND_CTRL	Output	Activates circuitry for front-end sensing

Table 5: Hardware I/O Table

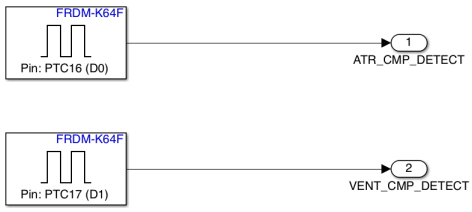


Figure 2: Hardware Input subsystem

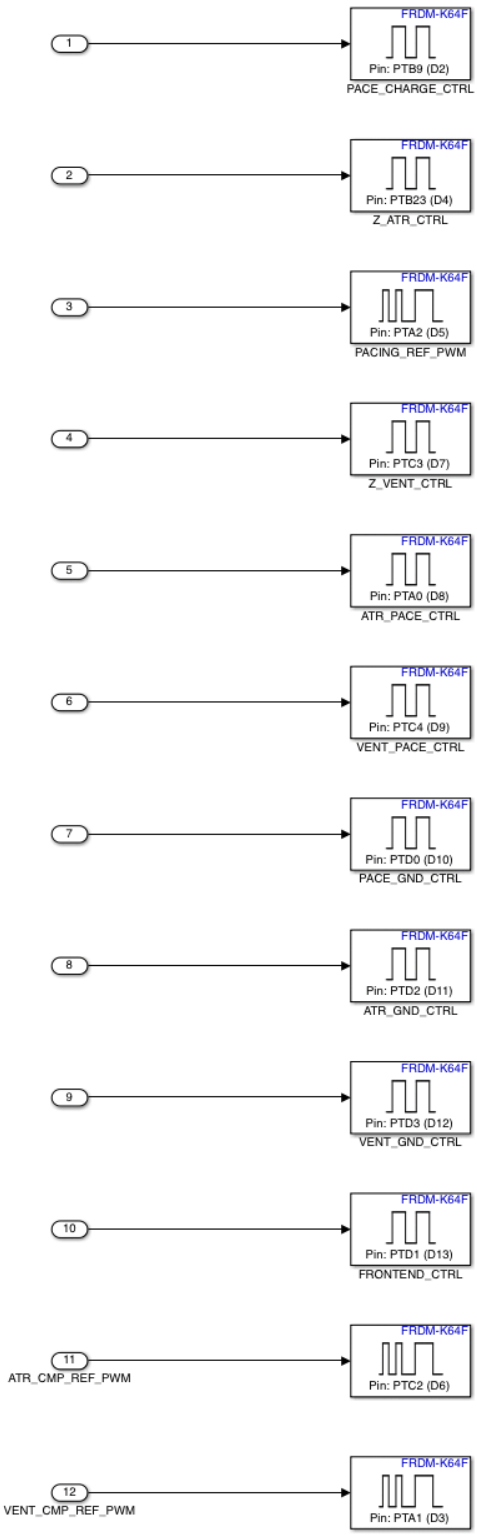


Figure 3: Hardware Output subsystem

2.3.4 State Machine Design

This state machine manages the pacemaker's mode decision-making logic. It focuses entirely on mode control and state transitions rather than handling any hardware inputs or outputs. The selection between the four pacing modes (AOO, VOO, AAI, and VVI), which operate within their own sub-state machines, is based on the MODE input. The system begins in a default state with FRONTEND_CTRL set low, disabling sensing until a mode is chosen. Each mode operates independently to ensure only one pacing mode is active at a time. Returning to the default state when the mode changes prevents overlap. This design provides a safe, modular, and scalable structure that allows clear transitions between modes while maintaining hardware isolation.

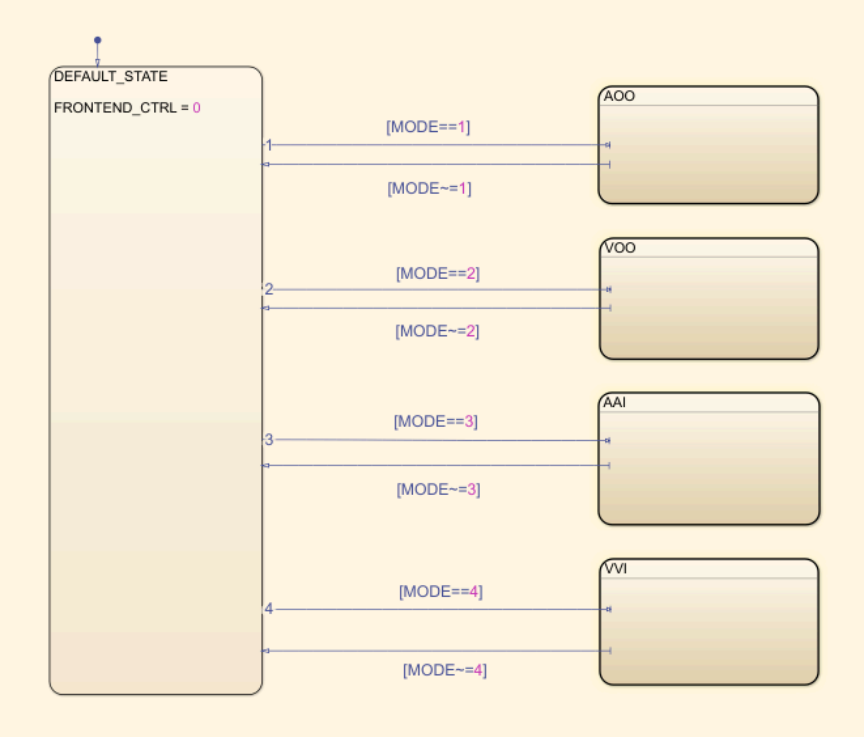


Figure 4: Bradycardia States subsystem

2.3.4.1 **Mode: AOO & VOO**

The AOO and VOO modes were designed as asynchronous pacing systems, meaning the pacemaker delivers electrical stimuli at a fixed interval, independent of the heart's intrinsic signals. This was achieved by implementing two main states. STATE_1 includes charging of the primary capacitor and discharging of the block capacitor. STATE_2 delivers atrial/ventricular pacing. The transition from STATE_1 to STATE_2 ensures that pacing occurs at regular intervals based on the programmed LRL, accounting for the programmed pulse width.

In AOO mode, only the atrium is paced, while in VOO, only the ventricle is paced. Both designs fulfill the requirement for asynchronous pacing by excluding sensing inputs, ensuring consistent pulse delivery regardless of intrinsic heart activity.

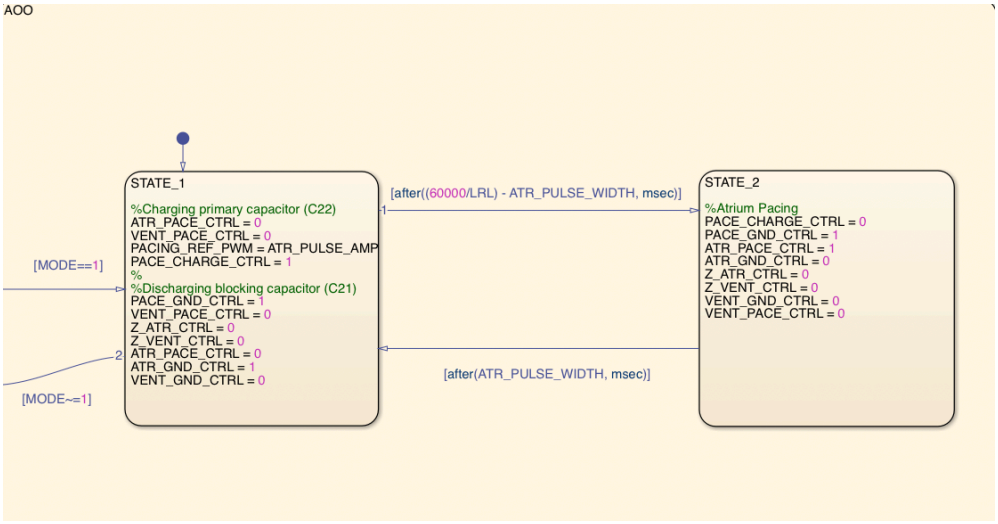


Figure 5: AOO Pacing Mode

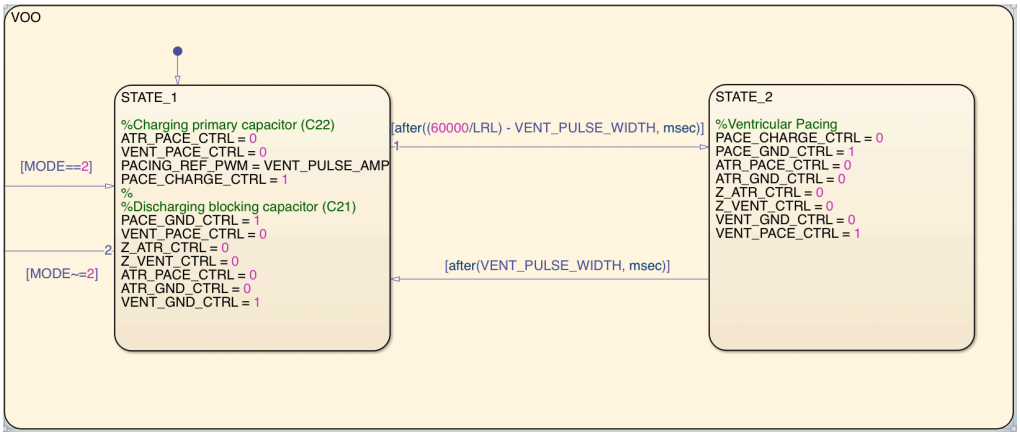


Figure 6: VOO Pacing Mode

2.3.4.2 Mode: AAI & VVI

The AAI and VVI modes include chamber sensing to allow the pacemaker to inhibit pacing when natural heart activity is detected. Similar to AOO and VOO modes, AAI and VVI consist of STATE_1 and STATE_2, which have the same behaviour as previously described. However, FRONTEND_CTRL is set to high to activate the sensing circuitry. A third state, the sensing state (ATR_SENSE or VENT_SENSE), was introduced to monitor the corresponding hardware input signal (ATR_CMP_DETECT or VENT_CMP_DETECT). The system waits through the set refractory period (ARP or VRP) before checking for new heart activity. This delay prevents the pacemaker from mistaking the pacing pulse or any electrical noise for a real heartbeat. These refractory periods are accounted for in the state transitions for accurate sensing.

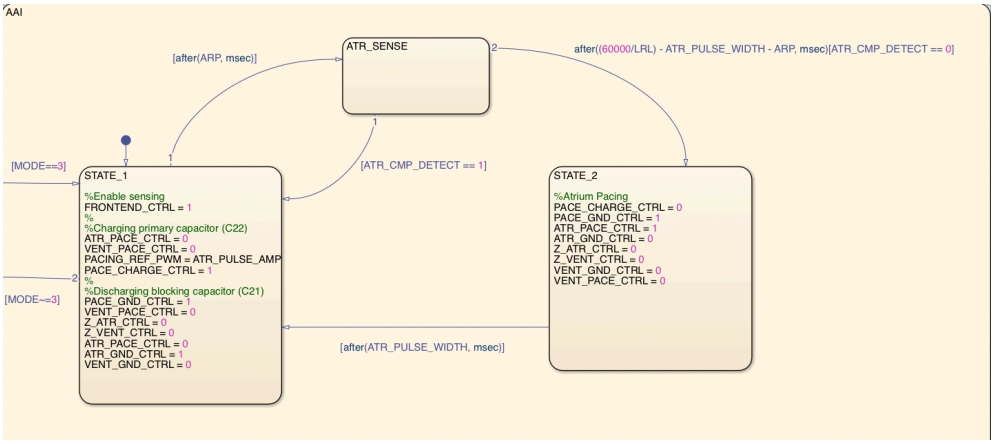


Figure 7: AAI Pacing Mode

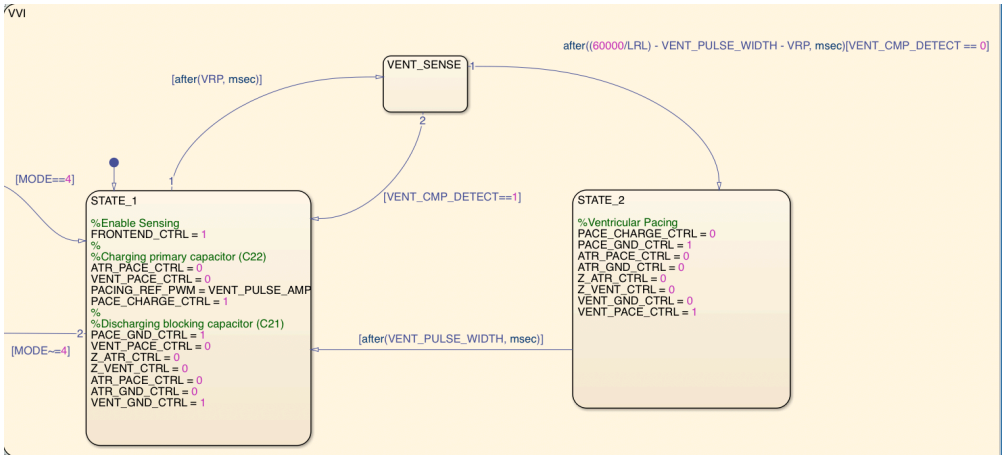


Figure 8: VVI Pacing Mode

2.3.5 Simulink Diagram

The programmable parameters and hardware inputs act as inputs to the pacemaker's main state machine, which contains sub-state machines for each pacing mode (AOO, VOO, AAI, and VVI). Each sub-state machine controls the sensing and pacing behaviour specific to its mode. Based on these inputs, the active pacing mode controls the hardware output signals, which drive the pacing circuitry. This structure ensures that parameter changes and sensed input signals directly influence pacing while keeping hardware control isolated from the core logic.

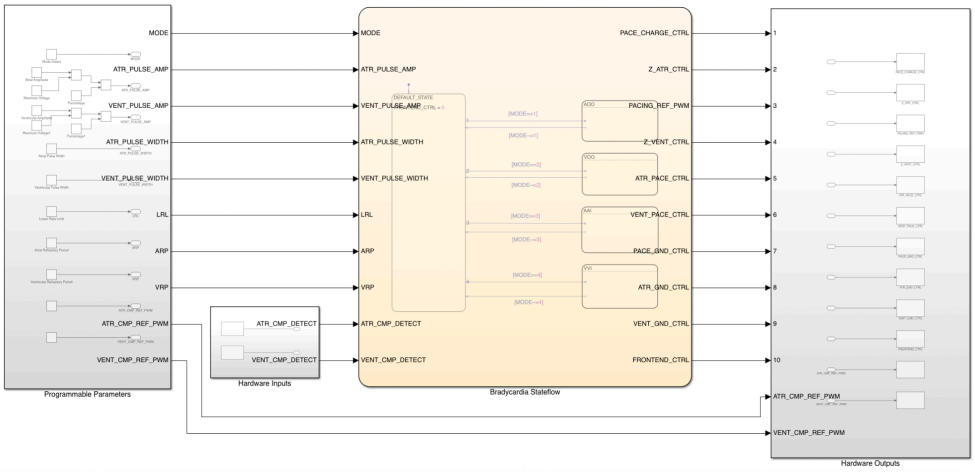


Figure 9: Pacemaker Architecture

2.3.6 DCM - Software Structure

The DCM application is implemented in Python 3.13 using Tkinter for the GUI. The system follows a modular design with clear separation between authentication, user interface, data management, and DICOM handling.

2.3.6.1 System Architecture (DCM)

```
PACEMAKER_DCM/
├── auth/              # Authentication module
├── data/              # User parameter storage
├── dicom/             # DICOM file handling
├── gui/               # GUI modules
│   ├── login.py
│   ├── main_interface.py
│   └── patient_select.py
├── requirements.txt   # Dependency list
└── main.py           # Entry point
```

Figure 10: DCM System Directory Structure

2.3.6.2 Core Modules

- Authentication Module (auth/auth.py): Manages user registration and login with *bcrypt* password hashing. Enforces 10-user limit through database constraints.
- Login Interface (gui/login.py): Provides authentication UI and launches patient selection on successful login. Masks password input for security.
- Patient Selection (gui/patient_select.py): Patient record management and DICOM initialization. Creates data structures for storing electrogram (EGM) waveforms.
- Main Interface (gui/main_interface.py): Primary control interface implementing all DCM requirements

2.3.6.3 User Interface Layout

Figure 11: Patient selection interface in the DCM

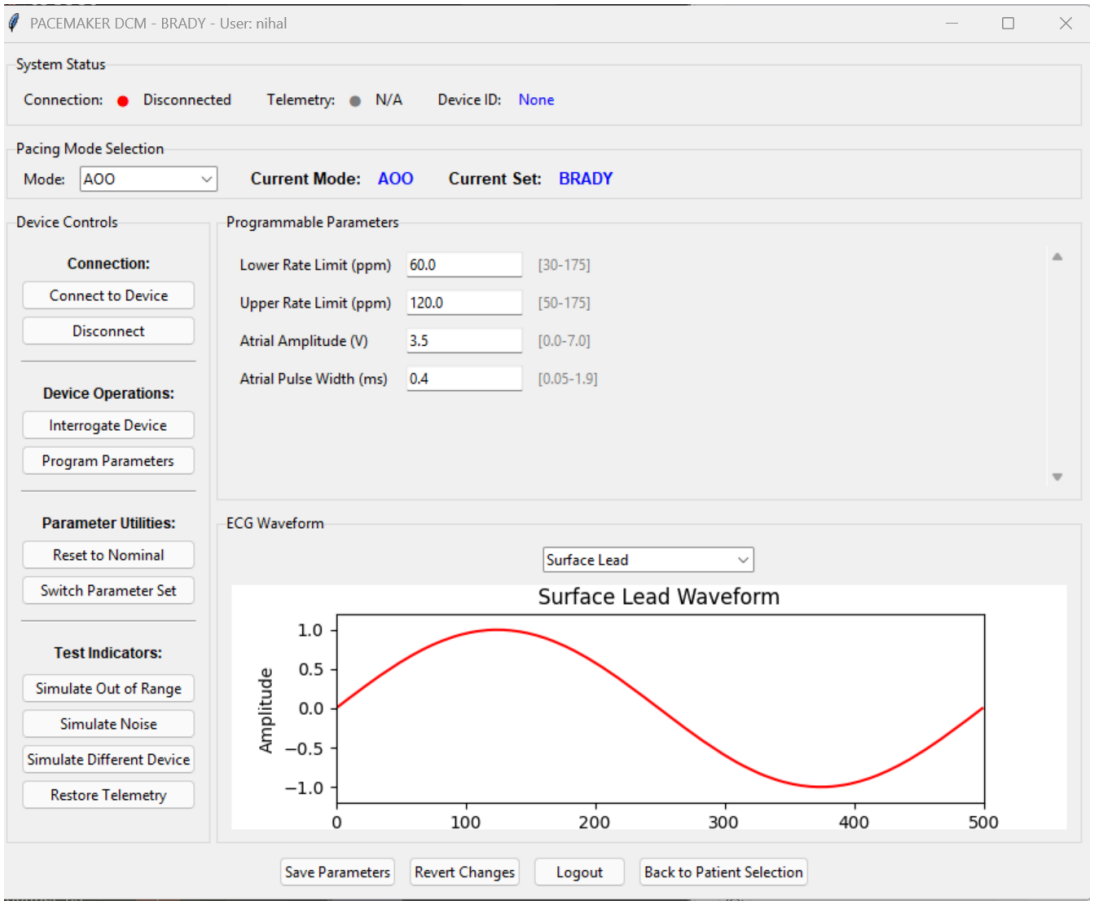


Figure 12: Main DCM interface with parameters and ECG waveform

The main interface (Figure 12) consists of:

- System Status Bar (top): Connection indicator (red/green), Telemetry status (green/orange/gray), Device ID, and Device Change Warning (Requirements 3.2.2.4-7)
- Mode Selection: Dropdown for AOO/VOO/AAI/VVI with current mode display
- Device Controls (left panel):
    - Connection: Connect/Disconnect buttons
    - Operations: Interrogate Device, Program Parameters
    - Utilities: Reset to Nominal, Switch Parameter Set
    - Test Indicators: Simulate Out of Range, Noise, Different Device, Restore Telemetry
- Programmable Parameters (center): Scrollable fields showing only mode-relevant parameters with valid ranges
- ECG Waveform Display (bottom): Dropdown selector (Surface/Atrial/Ventricular leads) with matplotlib visualization
- Action Buttons (bottom): Save Parameters, Revert Changes, Logout, Back to Patient Selection

2.3.6.4 **Data Flow**

- Authentication: User credentials → login.py → auth.py validates against users.db → Patient Selection
- Patient Selection: User selects/creates patient (Figure 11) → Initialize DICOM structures → Load patient-specific parameters → Main Interface
- Parameter Configuration: Select mode → Update display dynamically → Modify values → Validate → Save to data/params_<username>.json
- Device Programming (simulated): Connect → Validate parameters → Program (Figure 12 shows interface ready for serial communication in Deliverable 2)

2.3.6.5 **Design Principles**

- Information Hiding: Hardware abstracted from UI; database hidden behind auth API; validation encapsulated in class
- Modularity: Single responsibility per module; low coupling (GUI uses auth, not vice versa); high cohesion
- Robustness: Mode parameters in dictionaries (easily extensible); parameter ranges centralized; cross-platform file paths using os.path.join()
- Error Handling: Input validation before processing; try-catch for database operations; clear error messages; no inconsistent states

2.3.6.6 **Dependencies**

Package	**Purpose**
bcrypt	Password hashing
pydicom	DICOM format
numpy	Waveform arrays
matplotlib	ECG visualization
tzlocal	Timestamps
pillow	Image support

Table 6: Software Dependencies Used and Their Functions

- Built-in: tkinter, sqlite3, json, os, datetime
- Installation: pip install -r requirements.txt

3 Part 2

3.1 Requirements Potential Changes

3.1.1 Additional Pacing Modes

Four (4) additional pacing modes (AOOR, VOOR, AAIR, VVIR) will be added and configured.

3.1.2 Rate Adaptivity

Upcoming models are rate-adaptive versions of our previous models, requiring the incorporation of the accelerometer to automatically adjust the pacing rate based on the patient's activity level.

3.1.3 Serial Communication

Additionally, serial communication between the pacemaker and Device Controller-Monitor (DCM) will be configured. This will allow real-time data transfer to display signals, such as electrogram readings.

3.1.4 Additional Parameters

New parameters such as "Pulse Sensitivity" and "Width Tolerance" will be added to help make the system more precise.

3.2 Design Decision Potential Changes

Below design decisions may require modification in the next deliverable:

3.2.1 Device Communication Architecture

Current	Simulated device connection in main_interface.py
Potential Change	Separate serial communication module with a dedicated thread for non-blocking I/O
Reason	Real-time telemetry and device programming in Deliverable 2 require asynchronous communication to prevent UI freezing

Table 7: Device Communication Architecture

3.2.2 Waveform Display Implementation

Current	Static matplotlib plots embedded in Tkinter
Potential Change	Real-time streaming with a dynamic update mechanism
Reason	Live EGM display requires continuous data updates without recreating the entire plot

Table 8: Waveform Display Implementation

3.2.3 Mode-Specific Logic

Current	Dictionary-based MODE_PARAMETERS for parameter mapping
Potential Change	Object-oriented approach with mode classes inheriting from base pacing mode
Reason	Adding AOOR, VOOR, AAIR, and VVIR modes may benefit from an inheritance-based design for shared behaviours

Table 9: Mode-Specific Logic

3.2.4 Error Handling Strategy

Current	Message boxes for all errors
Potential Change	Logging system with error levels and a persistent error log file
Reason	Debugging production issues and regulatory compliance requires comprehensive error tracking

Table 10: Error Handling Strategy

3.3 Module Description

3.3.1 Module: Authorization Handling (auth/auth.py)

Purpose: Handles user authentication for the DCM application. Responsibilities include:

- Creating the SQLite user database
- Adding and verify user with hashed passwords
- Enforcing a maximum number of users (10)
- Clearing all users and associated data upon request

Public Functions:

Function Name	Description
init_db()	Creates the users table in SQLite database if it does not exist.
add_user (username, password)	Adds a new user with hashed password; raises error if maximum users reached or username exists.
check_login (username, password)	Verifies entered credentials against stored hashed password; returns True if valid.
clear_users()	Deletes all users from the database and removes all corresponding data folders.

Table 11: Authorization Handling Public Functions

Internal Functions:

Function Name	Description
get_user_count()	Returns the number of users currently registered.

Table 12: Authorization Handling Internal Functions

State / Global Variables:

Variable Name	Description
MAX_USERS	Maximum allowed users (10).
BASE_DIR	Base directory of the project.
DB_FILE	Path to SQLite database file.

Table 13: Authorization Handling State/Global Variables

Interactions:

- Files / Folders
    - Reads/writes users.db (SQLite database) in data/
    - Deletes all user folders under data/ when clearing users
- External Libraries
    - sqlite3 - Database management.
    - bcrypt - Password hashing and verification.
    - os / shutil - Directory and file operations.

3.3.2 Module: Login GUI (gui/login.py)

Purpose: Provides the user login and registration GUI for the DCM application. Responsibilities include:

- Allowing existing users to log in
- Registering new users
- Clearing all users (administrative function)
- Launching the patient selection GUI upon successful login

Public Functions:

Function Name	Description
main()	Initializes the login database and displays the login GUI.
attempt_login (user, pwd, root)	Verifies credentials and launches patient selection on success.
attempt_register (user, pwd)	Registers a new user if username is unique and max users not exceeded.
clearing_users()	Clears all users from the database after confirmation.

Table 14: Login GUI Public Functions

Internal Functions:

Function Name	Description
launch_patient_select (username)	Launches the PatientSelectApp GUI for a logged-in user.

Table 15: Login GUI Internal Functions

State / Global Variables

Variable	Description
root	Tkinter root window.
username_entry, password_entry	Entry fields for login credentials.
btn_frame, frame	Layout frames for buttons and inputs.
font_style	Font style and size used for labels and buttons.

Table 16: Login GUI State/Global Variables

Interactions

- Files / Folders
    - Interacts with “auth.py” functions, which read/write “users.db” and user folders under “data/”
- Other Modules
    - patient_select - On successful login, launches PatientSelectApp
- External Libraries

- tkinter - GUI
- messagebox - User feedback dialogs

3.3.3 **Module: DICOM Initialization (dicom/dicom_init.py)**

Purpose: Provides functions to create initial DICOM files for a patient. These files serve as templates for storing and manipulating DCM data in the application, such as:

- Patient information (SR modality)
- Bradycardia and temporary pacemaker parameters (SR modality)
- ECG waveforms for atrial/ventricular leads and surface ECG (ECG modality)

Public Functions:

Function Name	**Description**
patient_info_init (patientID, DCM_FILE)	Creates a DICOM file containing patient info and device metadata.
bradycardia_param_init (patientID, DCM_FILE)	Creates a DICOM file with bradycardia parameters for all pacing modes.
temporary_param_init (patientID, DCM_FILE)	Creates a DICOM file with temporary parameters for all pacing modes.
lead_waveform_init (patientID, DCM_FILE)	Creates a DICOM ECG file with atrial and ventricular lead waveforms.
surface_ecg_init (patientID, DCM_FILE)	Creates a DICOM ECG file with standard 12-lead surface ECG waveform data.

Table 17: DICOM Initialization Public Functions

Internal Functions:

Function Name	Description
param_sequence (mode)	Returns a DICOM Dataset representing all parameters for a specific pacing mode.

Table 18: DICOM Initialization Internal Functions

State / Global Variables

Variables	Description
PARAM_UNITS	Maps parameter names to their SI units.
MODE_PARAMETERS	Lists parameters for each pacing mode (AOO, VOO, AAI, VVI).

Table 19: DICOM Initialization State/Global Variables

Interactions

- Files / Folders
    - Writes DICOM files to DCM_FILE paths provided by caller
- External Libraries
    - pydicom – Dataset creation, file meta handling, saving DICOM files.
    - tzlocal / datetime – Local timezone timestamps for creation/modification.

3.3.4 Module: DICOM Modulation (dicom/dicom.py)

Purpose: Handles reading, writing, and initialization of patient-specific DICOM files for the DCM application, including:

- Creating necessary DICOM files for a new patient account
- Reading and writing parameter values DICOM files
- Reading and writing ECG waveform data from ECG DICOM files

Public Functions:

Function Name	Description
init_dir (username, patientID)	Creates a patient folder and initializes all required DICOM files. Returns paths to the files.
get_parameter (filepath, mode, parameter, unit_flag=False)	Returns the numeric value (or string with units) of a parameter from a DICOM SR file.

set_parameter (filepath, mode, parameter, value, save=True)	Sets a numeric parameter in a DICOM SR file and saves it by default (with the optional to not save).
get_waveparam (filepath, label, parameter)	Fetches a waveform parameter from a specified lead in an ECG DICOM file.
set_waveparam (filepath, label, parameter, value, save=True)	Sets a waveform parameter in a specified lead of an ECG DICOM file and saves it by default (with the optional to not save).
get_ecg_waveform (filepath, label)	Retrieves waveform data as a NumPy array for plotting.
set_ecg_waveform (filepath, label, data)	Writes waveform data to a specified lead in an ECG DICOM file, converting to standard int16 format.

Table 20: DICOM Modulation Public Functions

State / Global Variables

Variable	**Description**
None	N/A

Table 21: DICOM Initialization State/Global Variables

Interactions

- Files / Folders
    - Creates patient-specific folders in “data/{username}/{patientID}”
    - Reads and writes DICOM files (.dcm) for SR and ECG modalities
- Other Modules
    - dicom_init - Calls DICOM initialization functions to create new DICOM templates
- External Libraries
    - pydicom – Reading and writing DICOM files
    - numpy – Handling waveform data
    - datetime / tzlocal – Timestamp management

- os – Directory management

3.3.5 **Module: Patient Selection GUI (gui/patient_select.py)**

Purpose: Provides a GUI for users to manage patients in the DCM system. Responsibilities include:

- Listing existing patients for a user
- Adding new patients and initializing their DICOM files with default data
- Removing patients and cleaning up associated DICOM files
- Launching the main DCM interface for a selected patient

Public Functions:

Function Name	**Description**
__init__(self, root, username)	Initializes the patient selection GUI and loads existing patient records.
refresh_list(self)	Updates the listbox to show current patients and manages button states.
proceed(self)	Launches DCMMainInterface for the selected patient.
add_patient(self)	Prompts for patient information, initializes DICOM files, sets default parameters, and updates patient to user JSON.
remove_patient(self)	Removes the selected patient from JSON and deletes their folder.

Table 22: Patient Selection GUI Public Functions

Internal Functions:

Function Name	**Description**
generate_patient_id (existing_ids)	Generates a unique 5-digit patient ID.
default_parameters (paths)	Sets default bradycardia and temporary parameters from “default_params.json” to DICOM files.

Table 23: Patient Selection GUI Internal Functions

State / Global Variables

Variable	Description
username	Current user identifier.
user_dir	Path to the user's patient data directory.
patients_file	Path to JSON file storing patient records.
patients_data	Loaded patient records as a dictionary.
selected_patient_id	Currently selected patient ID from the listbox
Tkinter widgets	[root, listbox, proceed_btn, add_btn, remove_btn]

Table 24: Patient Selection GUI State/Global Variables

Interactions

- Files / Folders
    - Reads/writes "patients.json" in the user folder
    - Creates/deletes patient folders under "data/{username}/{patientID}"
    - Reads/writes DICOM files via "dicom.py"
    - Loads default parameters from "default_params.json"
- Other Modules
    - dicom - File initialization and parameter management
    - DCMMainInterface - Launched for a selected patient
- External Libraries
    - pydicom - Reading and updating DICOM datasets
    - tkinter – GUI
    - numpy - Generating example ECG waveforms
    - random - Unique patient ID generation

3.3.6 **Module: Main GUI (gui/main_interface.py)**

Purpose: Provides the main graphical user interface for the DCM system. Responsibilities include:

- Selecting pacing modes and parameter sets (Bradycardia or Temporary)
- Displaying and editing programmable pacemaker parameters
- Connecting to and interacting with a simulated pacemaker device
- Displaying ECG waveforms for atrial, ventricular, and surface leads
- Saving, reverting, or resetting parameters
- Managing user session

Public Functions:

Function Name	**Description**
__init__ (self, root, username, patientID)	Initializes GUI, patient directories, JSON/DICOM data, and sets default mode/lead.
on_mode_change (self, event=None)	Handles user mode selection change; updates parameters display.
on_lead_change(self, event=None)	Handles ECG lead selection change and re-plots waveform.
switch_parameter_set (self)	Switches between Bradycardia and Temporary parameter sets.
save_parameters(self)	Validates and saves current parameters to DICOM and JSON.
revert_changes(self)	Reverts parameters to last saved values.
reset_to_nominal(self)	Resets current mode parameters to nominal default values.
logout(self)	Logs out the user and returns to the login screen.
back_to_patient_selection (self)	Returns to patient selection GUI.
Device Control (Simulated)	
connect_device(self)	Simulates connecting to a device.
disconnect_device (self)	Disconnects from simulated device.
GUI Display	
create_main_interface (self)	Builds all GUI components.
plot_waveform(self)	Plots numeric waveform data from DICOM.
Internal Functions	
initialize_json_files (self)	Loads or creates JSON parameter files for Bradycardia and Temporary sets.

load_or_create_json (self, json_path, dicom_key)	Helper to initialize JSON from DICOM if missing.
validate_parameter (self, param_key, value)	Validates a single parameter against allowed range.

Table 25: Main GUI Public Functions

State / Global Variables

Variable	Description
username, patientID	Current user and patient identifiers.
patient_dir	Path to patient's DCM data folder.
current_mode, current_param_type, lead_type	Current selections.
PARAMETER_RANGES, MODE_PARAMETERS, PARAMETER_LABELS	Parameter metadata and validation ranges.
current_parameters	Editable parameters in the current mode.
current_json_data	JSON data for the current parameter set.
json_path, current_dcm_path	File paths for JSON and DICOM storage.
connected_device, last_device, connection_status	Device simulation state
tkinter widgets	[root, parameter_entries, parameter_widgets, mode_var, lead_var, fig, ax, canvas]
tkinter status labels	[connection_indicator, connection_text, telemetry_indicator, telemetry_text, device_label, device_warning]

Table 26: Main GUI State/Global Variables

Interactions

- Files / Folders
    - Reads/writes JSON parameter files
    - Reads/writes DICOM files via "dicom.py"
- Other Modules
    - login - Returns user to login on logout
    - patient_select - Returns to patient selection GUI

- External Libraries
    - tkinter - GUI
    - matplotlib - ECG plotting
    - numpy - Waveform data processing

3.4 Testing

3.4.1 Pacemaker - Simulink Model Testing

3.4.1.1 AOO & VOO Testing

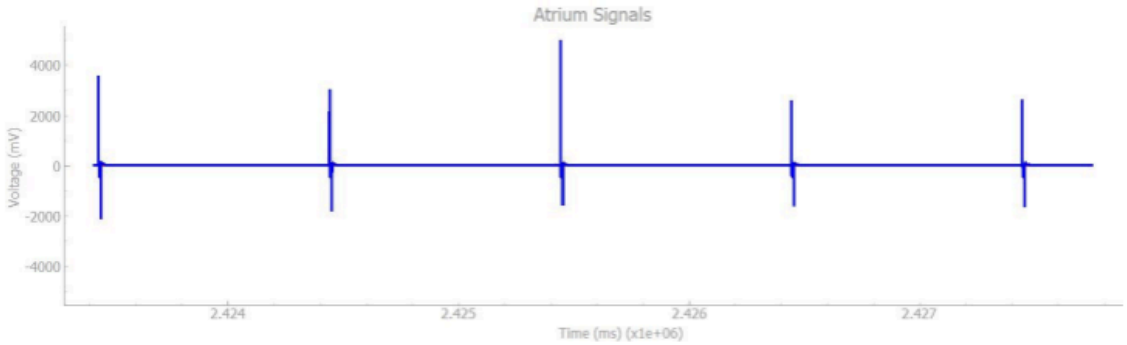
Both AOO and VOO share similar stateflow structures; thus, similar test cases were performed for each mode. The following tables outline various test cases conducted to verify the behaviour of AOO and VOO modes, ensuring the correct chambers are being paced independent of intrinsic heart activity. Pacemaker and Heartview parameters were set to values specified in each of the following tables.

Test case 1 was conducted to verify that the pacemaker produces periodic, regular atrial pulses, even when no intrinsic heart activity is present. This verifies that the mode is pacing the atrium without sensing and responding to atrial activity.

Pacemaker Settings		**Heartview Settings**		**Expected Output**	**Pass/Fail**
Mode	AOO	**Natural Atrium (ms)**	Off	The atrium should be paced at a steady rate of 60 BPM, without responding to intrinsic atrial or ventricular activity. Operates asynchronously and does not inhibit or synchronize with natural heart signals.	Pass
Pulse Width (ms)	10	**Natural Ventricle (ms)**	Off		
Pulse Amplitude (V)	3.5	**Natural Heart Rate (BPM)**	30		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 27: Test Case 1 (AOO)

Atrium Plot:



Ventricle Plot:

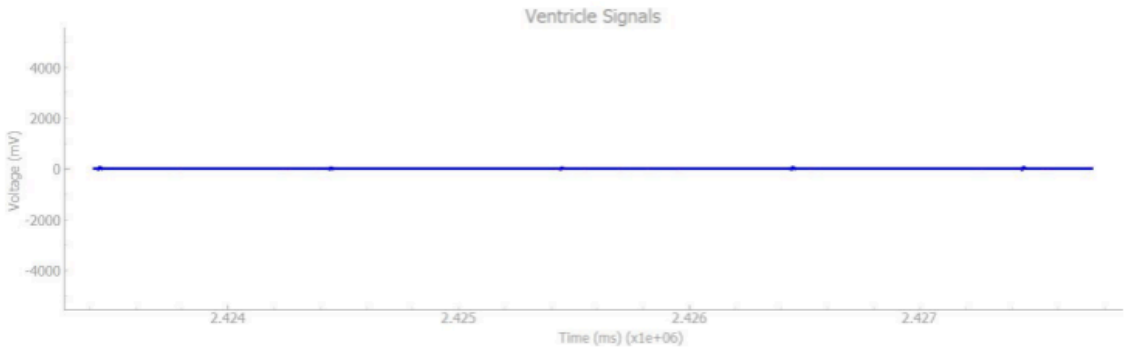


Figure 13: Test Case 1 Results

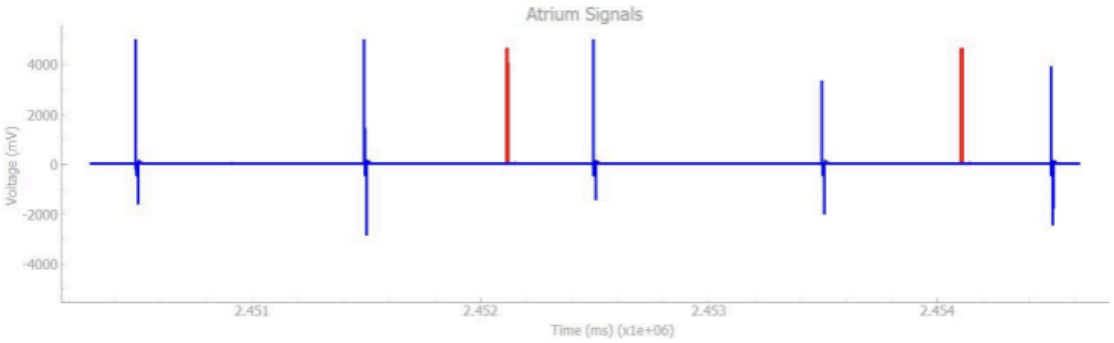
As seen in the Figure above, no intrinsic heart signal is visible, but the pacemaker delivers regular pacing pulses to only the atrium. The pulses are delivered every 1000ms, verifying the atrium is being paced at 60 BPM.

Test case 2 was conducted to verify that the pacemaker continues to asynchronously deliver periodic atrial pulses in the presence of intrinsic heart activity. The natural heart rate was set to 30 BPM to simulate a bradycardic condition. This test case verifies that the pacemaker operates without affecting the natural signals.

Pacemaker Settings		Heartview Settings		Expected Output	Pass/Fail
Mode	AOO	**Natural Atrium (ms)**	5	The atrium should be paced at a steady rate of 60 BPM. The pacemaker does not respond to intrinsic atrial or ventricular activity, operating asynchronously. Natural signals should remain unaffected by the pacing.	Pass
Pulse Width (ms)	10	**Natural Ventricle (ms)**	5		
Pulse Amplitude (V)	3.5	**Natural Heart Rate (BPM)**	30		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 28: Test Case 2 (AOO)

Atrium Plot:



Ventricle Plot:

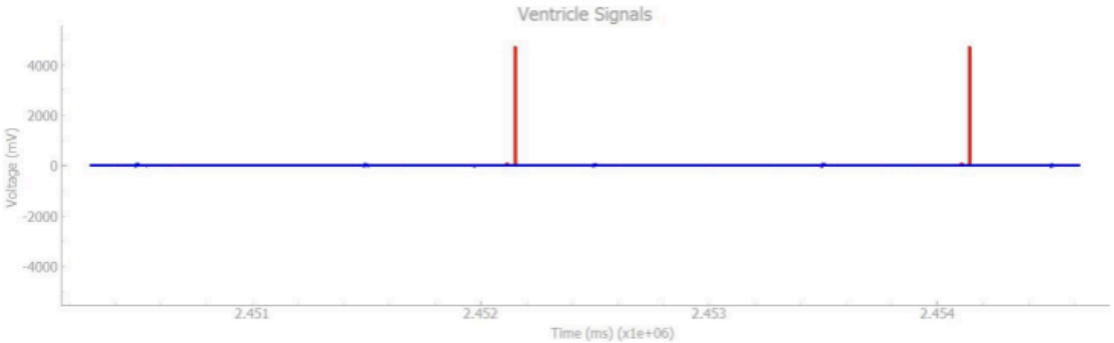


Figure 14: Test Case 2 Results

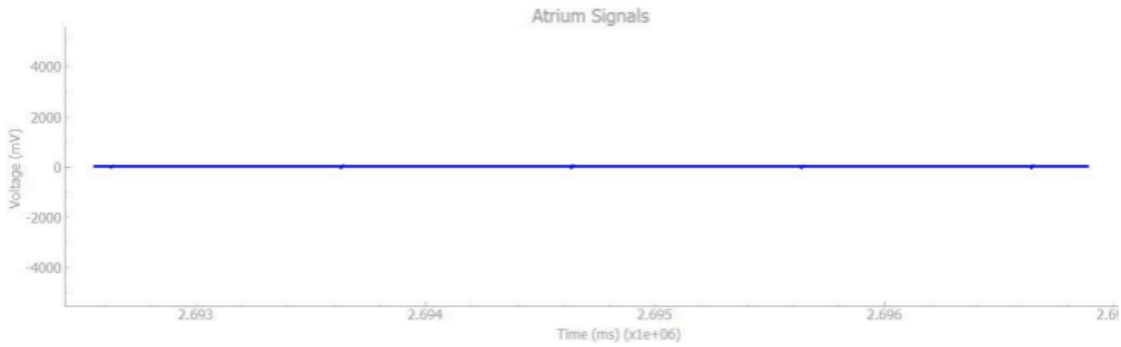
The above figure shows consistent atrial pacing occurring every 1000 ms, corresponding to a rate of 60 BPM, while the ventricular plot remains flat with no pacing activity. Both plots show that the natural heart rate at 30 BPM is unaffected by the pacing. This confirms that the pacemaker correctly stimulates only the atrium in AOO mode and operates asynchronously, independent of intrinsic heart signals.

Similar to test case 1, test case 3 aimed to verify the same behaviour except with respect to the ventricle rather than the atrium.

Pacemaker Settings		**Heartview Settings**		**Expected Output**	**Pass/Fail**
Mode	VOO	**Natural Atrium (ms)**	Off	The ventricle should be paced at a steady rate of 60 BPM, without responding to intrinsic atrial or ventricular activity. Should operate asynchronously and does not inhibit or synchronize with natural heart signals.	Pass
Pulse Width (ms)	10	**Natural Ventricle (ms)**	Off		
Pulse Amplitude (V)	3.5	**Natural Heart Rate (BPM)**	30		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 29: Test Case 3 (VOO)

Atrium Plot:



Ventricle Plot:

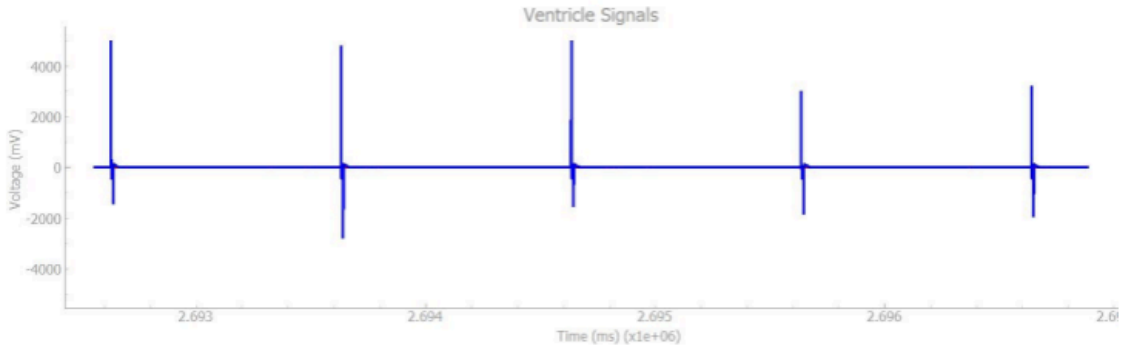


Figure 15: Test Case 3 Results

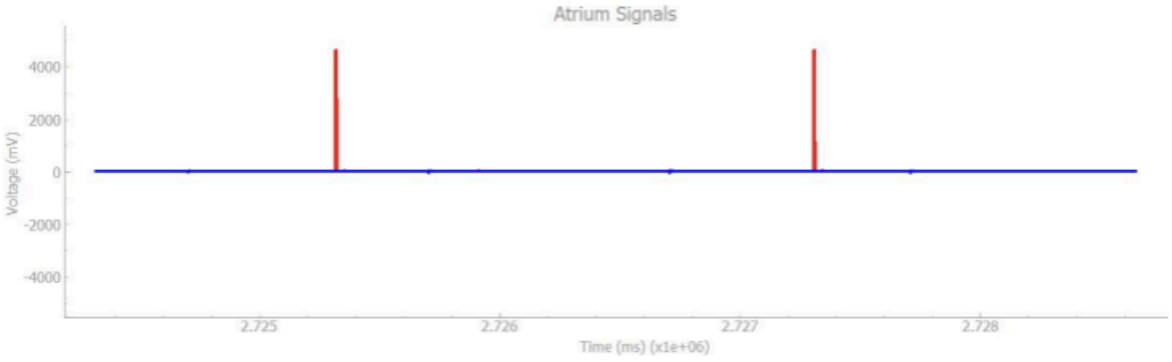
As seen in the Figure above, no intrinsic heart signal is visible, but the pacemaker delivers regular pacing pulses to only the ventricle. The pulses are delivered every 1000ms, verifying the ventricle is being paced at 60 BPM.

Similar to test case 2, test case 4 aimed to verify the same behaviour except with respect to the ventricle rather than the atrium.

Pacemaker Settings		Heartview Settings		Expected Output	Pass/Fail
Mode	VOO	**Natural Atrium (ms)**	5	The ventricle should be paced at a steady rate of 60 BPM. The pacemaker does not respond to intrinsic atrial or ventricular activity, operating asynchronously. Natural signals should remain unaffected by the pacing.	Pass
Pulse Width (ms)	10	**Natural Ventricle (ms)**	5		
Pulse Amplitude (V)	3.5	**Natural Heart Rate (BPM)**	30		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 30: Test Case 4 (VOO)

Atrium Plot:



Ventricle Plot:

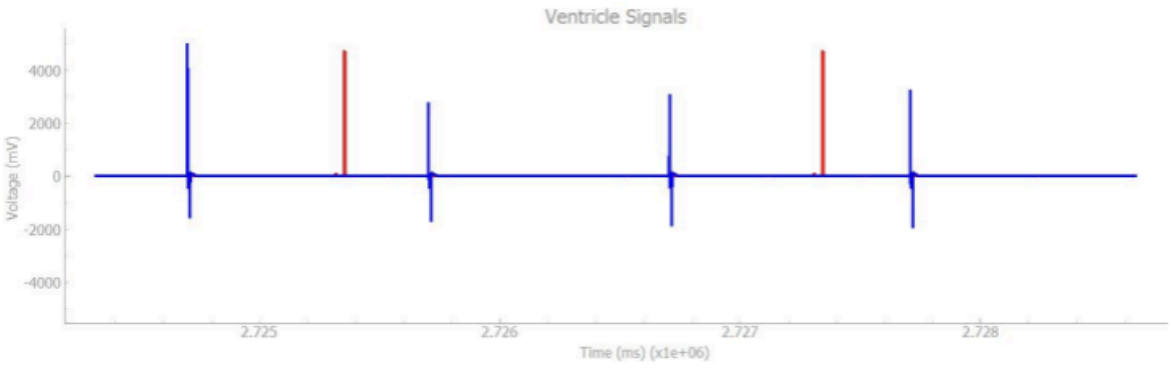


Figure 16: Test Case 4 Results

The above figure shows consistent ventricular pacing occurring every 1000 ms, corresponding to a rate of 60 BPM, while the atrial plot remains flat with no pacing activity. Both plots show that the natural heart rate at 30 BPM is unaffected by the pacing. This confirms that the pacemaker correctly stimulates only the atrium in AOO mode and operates asynchronously, independent of intrinsic heart signals.

3.4.1.2 AAI & VVI Testing

Test case 5 aims to verify AAI mode behaviour when the natural heart rate is less than the lower rate limit. This test confirms that the pacemaker correctly senses natural atrial activity and delivers atrial pacing pulses accordingly to maintain the minimum rate under a bradycardic condition.

Pacemaker Settings		**Heartview Settings**		**Expected Output**	**Pass/Fail**
Mode	AAI	**Natural Atrium (ms)**	5	The natural atrial pulse overrides the pacemaker every other beat, thus, inhibiting pacing when a natural signal occurs.	N/A*
Pulse Width (ms)	10	**Natural Ventricle (ms)**	Off		
Pulse Amplitude (V)	4.5	**Natural Heart Rate (BPM)**	30		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 31: Test Case 5 (AAI)

Test case 6 aims to verify AAI mode behaviour when the natural heart rate is equal to the lower rate limit.

Pacemaker Settings		**Heartview Settings**		**Expected Output**	**Pass/Fail**
Mode	AAI	**Natural Atrium (ms)**	5	The natural atrial pulse overrides the pacemaker for every beat, thus inhibiting pacing at all times since	N/A*

Pulse Width (ms)	10	**Natural Ventricle (ms)**	Off	the natural heart rate meets LRL.	
Pulse Amplitude (V)	4.5	**Natural Heart Rate (BPM)**	60		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 32: Test Case (AAI)

Test case 7 aims to verify similar behaviour as test case 5, but with respect to the ventricle and VVI mode.

Pacemaker Settings		**Heartview Settings**		**Expected Output**	**Pass/Fail**
Mode	VVI	**Natural Atrium (ms)**	Off	The natural ventricular pulse overrides the pacemaker every other beat, thus, inhibiting pacing when a natural signal occurs.	N/A*
Pulse Width (ms)	10	**Natural Ventricle (ms)**	5		
Pulse Amplitude (V)	4.5	**Natural Heart Rate (BPM)**	30		
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 33: Test Case 7 (VVI)

Test case 8 aims to verify similar behaviour as test case 6, but with respect to the ventricle and VVI mode.

Pacemaker Settings		**Heartview Settings**		**Expected Output**	**Pass/Fail**
Mode	VVI	**Natural Atrium (ms)**	Off	The natural ventricular pulse overrides the pacemaker for every beat, thus inhibiting pacing at all times since the natural heart rate meets LRL.	N/A*
Pulse Width (ms)	10	**Natural Ventricle (ms)**	5		
Pulse Amplitude (V)	4.5	**Natural Heart Rate (BPM)**	60		

Pacemaker Settings		Heartview Settings		Expected Output	Pass/Fail
Lower Rate (BPM)	60	**Natural AV Delay (ms)**	30		

Table 34: Test Case 8 (VVI)

***NOTE**: Actual outcomes were not obtained for AAI and VVI modes, as the NXP FRDM-K64F stopped functioning. The board's OpenSDA port stopped working; thus, the pacemaker was not recognizable by a computer, and testing could not be completed. Troubleshooting steps were taken, and power-cycling was attempted, but all efforts failed to recover the board.*

3.4.1.3 Authorization Handling and Login GUI

Testing for the authentication and login modules focuses on verifying user credential handling, database integrity, and GUI flow. Testing confirms that the database is correctly initialized with the expected values and that user registration enforces constraints such as unique usernames and the ten-user limit. Passwords are correctly validated against their hashed equivalents. The GUI should display appropriate success or error messages and correctly transition between login/registration, and patient selection interfaces.

3.4.1.4 dDICOM Initialization and DICOM Modulation

Testing for the DICOM initialization and handling modules ensures that DICOM files are created, read, and modified accurately. Each file should be verified to contain required metadata and the correct modality tags. Unit tests should confirm that functions correctly access and update stored parameter values, while waveform data handling and saving functions should maintain data integrity and format consistency. Edge cases such as missing tags, empty datasets, or invalid mode names are handled with clear error feedback.

3.4.1.5 **Patient Selection GUI**

Testing for the patient selection module verifies patient record management and correct DICOM directory setup. Adding a patient should create a new, properly structured folder under the user's directory, initialize all required DICOM files, and populate them with patient information and waveform data. Validation tests should ensure that input fields such as birthdate and sex enforce correct formats and accepted values. Removing a patient should cleanly update the local JSON record and delete the associated folder and DICOM files. GUI testing should confirm that the listbox updates dynamically when patients are added or removed. Finally, proceeding to the main interface must pass the correct username and patient ID.

3.4.1.6 **Main GUI**

Testing for the main DCM interface requires correct synchronization between GUI input, displayed waveform data, and stored DICOM values. Parameters shown should reflect the selected pacing mode and parameter set, while changes made through user input must update both the interface and corresponding DICOM files consistently. Waveform plot is tested with known signals (sine waves) to verify correct scaling between the DICOM file and the interface, and refresh behavior when switching leads or modes. Tests should also verify that saving, reverting, and resetting functions maintain data integrity. Overall, the interface should remain stable under repeated mode changes and parameter updates.

3.5 GenAI Usage

3.5.1 Simulink

No GenAI tool was used in the development of the Simulink side of things.

3.5.2 DCM

GenAI tools (Claude 3.5 Sonnet, GPT-5) were used to research appropriate Python dependencies (bcrypt, pydicom, matplotlib) and their purposes, apply new efficient techniques, troubleshoot implementation issues such as ModuleNotFoundError exceptions and cross-platform file path compatibility, and assist with structuring documentation.

3.5.3 Documentation

In terms of documentation, GenAI tools (GPT-5) were used to assist in naming tables and figures in a concise and informative manner, clarifying certain wording related to the pacemaker documentation, and formatting the overall document.

4 References

[1] Boston Scientific, PACEMAKER System Specification, Jan. 3, 2007. [Online]. Available: https://avenue.cllmcmaster.ca/d2l/le/content/712745/viewContent/5306027/View

[2] “pydicom documentation — pydicom 3.0.1 documentation.” Accessed: Oct. 27, 2025. [Online]. Available: https://pydicom.github.io/pydicom/stable/index.html

[3] “I.4 Media Storage Standard SOP Classes.” Accessed: Oct. 27, 2025. [Online]. Available: https://dicom.nema.org/dicom/2013/output/chtml/part04/sect_I.4.html

[4] “DICOM Library - Anonymize, Share, View DICOM files ONLINE.” Accessed: Oct. 27, 2025. [Online]. Available: https://www.dicomlibrary.com/dicom/dicom-tags/

[5] *tzlocal: tzinfo object for the local timezone*. Python.

[6] “datetime — Basic date and time types,” Python documentation. Accessed: Oct. 27, 2025. [Online]. Available: https://docs.python.org/3/library/datetime.html

[7] C. Xie, L. McCullum, A. Johnson, T. Pollard, B. Gow, and B. Moody, *Waveform Database Software Package (WFDB) for Python*. PhysioNet. doi: 10.13026/9NJX-6322.

[8] L. Maršánová, A. Nemcova, R. Smisek, L. Smital, and M. Vitek, “Brno University of Technology ECG Signal Database with Annotations of P Wave (BUT PDB).” PhysioNet. doi: 10.13026/HWVJ-5B53.

[9] “Add Salt to Hashing: A Better Way to Store Passwords | Auth0,” Auth0 - Blog. Accessed: Oct. 27, 2025. [Online]. Available: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/

[10] “SQL Tutorial.” Accessed: Oct. 27, 2025. [Online]. Available: https://www.w3schools.com/sql/

[11] “Python getattr() Function.” Accessed: Oct. 27, 2025. [Online]. Available: https://www.w3schools.com/python/ref_func_getattr.asp

[12] “Using PyInstaller — PyInstaller 6.16.0 documentation.” Accessed: Oct. 27, 2025. [Online]. Available: https://pyinstaller.org/en/stable/usage.html

[13] “Graphical user interfaces with Tk,” Python documentation. Accessed: Oct. 27, 2025. [Online]. Available: https://docs.python.org/3/library/tk.html

[14] “Hashing Passwords in Python with BCrypt,” GeeksforGeeks. Accessed: Oct. 27, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/hashing-passwords-in-python-with-bcrypt/

[15] R. Python, “Python GUI Programming: Your Tkinter Tutorial – Real Python.” Accessed: Oct. 27, 2025. [Online]. Available: https://realpython.com/python-gui-tkinter/

[16] “sqlite3 — DB-API 2.0 interface for SQLite databases,” Python documentation. Accessed: Oct. 27, 2025. [Online]. Available: https://docs.python.org/3/library/sqlite3.html

[17] “os.path — Common pathname manipulations,” Python documentation. Accessed: Oct. 27, 2025. [Online]. Available: https://docs.python.org/3/library/os.path.html