Document Title **Software IP Submission Form**		Version Dated 27-Apr-2025	Institute for Infocomm Research A*STAR
Owner of Form **SWIP Admin/RCMO**	SWIP ID Reference: (to be issued by SWIP coordinator) **SWIP-**	Version No. V1.12	

SOFTWARE IP SUBMISSION FORM

INFORMATION FOR COMPLETING THE SOFTWARE IP SUBMISSION FORM

1. Software IP (SWIP) creators or Software Developers summarize the key information/concept of the software generated from the work done at I^{2}R. This information will be treated as confidential and will be used to justify for subsequent exploitation of the software.
2. Subject matter in each software should be described clearly outlining the technical merits, usefulness, and applications of the software.
3. Software IP (SWIP) bank is repository of software codes developed during a course of work, that could be reused as fit for use software. SWIP may be derived from associated TDs (Technology Disclosure) / Patents or from project(s). Engineering effort is essential to create reusable software. It should be made sure the the SWIP, from a project, has not violated any legal terms (example: if a project has contractual obligation that the FIP (Foreground IP) belongs to the collaborator and can not be used in other projects, then that SWIP can not be submitted). However if a software IP is exclusively licensed and there is restriction of use of software developed under the contract, then the SWIP cannot or might not be reused within the stipulated duration set forth in the contract.
4. Please DO NOT submit the duplicate copy of project archival or Prototype archived which was submitted to PMO. SWIP is reusable software NOT the project archival and should not have project specific customer data / confidential information.
5. SWIP source code/ documents/library/Executable should be ready to upload in gitlab within 14 working days upon submitting the form. This is to plan accordingly the review meeting of the SWIP submitted. SWIP creator(s) must make sure uploading in gitlab of the complete code, libraries, documents which can be built, installed and run as working software.
6. SWIP is accounted on FY basis for KPI counting. However internally I^{2}R keeps track of SWIPs as CY (Calendar Year) basis for Depatment/Unit achievement records. **The last date for submission of SWIP, in a calender year, is by the 10th of December**. Submitter shall plan accordingly to submit the SWIP to be counted for the said CY. SWIP is counted in the CY upon review and approval by the review panel.
7. Upon approval of the SWIP by reviewers, this submit form is then emailed to A*ccelerate OIC to record as the approved SWIP. Evaluation records, approved by reviewers of the SWIP, are maintained internally by SWIP coordinator.

Please submit the original, completed Software IP submission form, in MS word and in pdf form, to:

Qu Xuhong, SWIP Coordinator

quxh@i2r.a-star.edu.sg, DID: 64082477

Software IP Submission Form

1. TITLE OF SOFTWARE IP *(a short but sufficiently descriptive title to identify the general nature of the software.)*

An Application for Semiconductor Process Data Management, Model Training, and Optimization

2. DESCRIPTION OF THE SOFTWARE

Please provide as complete a description as possible. This is essential for:
A. The purpose of your description is to enable a person with similar skills in your field to be able to make and use the software you submitted.
B. Please do not withhold any key elements of the software submitted

This application provides a unified interface for semiconductor process data management, predictive model training, and finding optimal process parameters for target performance. It efficiently handles experimental data, supports predictive modeling and inference, and provides inverse modeling functionality—recommending optimal experimental settings based on target outcomes.

The backend dynamically manages data from either manual entry or logs in Excel or CSV file formats, applies validation rules, and integrates AI-driven models to deliver statistical analyses and spatial visualizations. The intuitive frontend offers interactive data grids with editing, undo/redo, and bulk export features, dynamic model selection, and real-time predictions and training status updates.

Configuration of data inputs, UI, and models are managed externally via JSON and Excel files, enabling easy addition of new processes, datasets, and analytical methods without modifying core code.

2.1 Field of The Software:

A sentence or paragraph identifying the general ***field of technology*** *to which the software relates, which* ***Department/unit/clusters*** *could potentially reuse this software? How the problem was previously solved?*

The software targets the semiconductor process-development domain, applying AI-driven prediction and DOE management for different processes. It is directly reusable by A*STAR research units engaged in process optimization and advanced manufacturing, notably the Institute for Infocomm Research (I^2R) for AI and data-analytics integration, as well as the Institute of Microelectronics (IME), the Institute of Materials Research and Engineering (IMRE), the Singapore Institute of Manufacturing Technology (SIMTech), and Institute of High Performance Computing (IHPC) for large-scale simulation and computing support. Traditionally, semiconductor process development has relied on iterative, physics-based TCAD simulations and manual DOE cycles—an approach that is time- and resource-intensive. This software unifies physics-informed simulation data with machine-learning models to shorten development cycles, lower cost, and improve predictive accuracy.

2.2 Brief summary of the software:

A brief paragraph (similar to the abstract of a TD) describing the key feature(s) of the software with some background context. What is its core?

This software provides an integrated application for efficiently managing experimental parameters and performing AI-driven predictions in semiconductor manufacturing processes. It seamlessly integrates data from Excel and CSV files, applies predictive modelling to calculate essential statistical metrics, and generates spatial visualizations. The application also provides advanced inverse modelling capabilities—identifying optimal experimental conditions to achieve specified outcomes.

The interactive platform features editable data grids with built-in validation, undo/redo functionality, export options, dynamic model selection, and real-time prediction results. By combining experimental and simulation data with machine-learning inference, the software significantly accelerates process optimization and improves predictive accuracy, reducing both development time and resource expenditure.

2.3 **Detailed Description of the Software:**

This section should be detailed enough for a person having ordinary skill in your technical field to construct and use the software you describe.

(i) A detailed description of the software with key technical features, how it works and to give example of usage scenarios, where possible.

This platform is structured as a Flask-based application providing RESTful endpoints for experiment management, data import/export, predictive modelling, and training orchestration. Upon initialization, the backend caches experimental data (including source and response parameters) from Excel or CSV files, making it readily available for rapid analysis and display it as a table within the application interface.

Key Technical Features:

1. **Data Management**:
   - Data Loading: Endpoints parse data files, cache structured data for immediate access.
   - Editable Interface: Interactive data grids with cell-level editing, validation, undo/redo, and full-table updates.
   - Validation: Dynamic validation rules provide immediate feedback and enforce constraints.
2. **Predictive Modelling**:
   - Forward Prediction: Applies configurable machine learning models (e.g., neural networks, serialized models) to compute key metrics (average, non-uniformity, standard deviation, skewness) and generate spatial heatmaps.
   - Inverse Prediction: Given desired response parameters, the platform identifies and recommends source parameters that best achieve those targets, employing a similar interactive interface as forward prediction.
3. **Training Control**:
   - Training Execution: Initiates external model training processes, monitors status, and provides real-time UI updates.
4. **Data Export and Visualization:**
   - Export Functionality: Supports exporting tables and predictions in various formats, embedding visualizations directly within outputs.
   - Image Management: Generates and manages visual outputs of predicted and experimental outcomes.

Usage Scenario: Engineers select and edit experimental parameters within an interactive UI, validate changes, execute forward or inverse modelling, and immediately review results through intuitive metrics and visualizations. If model refinement is necessary, engineers initiate retraining directly from the interface, seamlessly monitoring the process.

This modular, extensible architecture enables straightforward deployment, customization, and extension by users familiar with Python, Flask, and standard machine-learning techniques.

(ii) Does your software possess any of its disadvantages or limitations? Can they be overcome?

1. **In-memory caching only**

- Limitation: All DOE tables and edits reside in RAM and are lost on server restart; memory usage grows with table size and user count.
- Solution: Persist caches in Redis or a relational database (e.g. PostgreSQL) to survive restarts, share across instances, and throttle memory growth.

2. **Blocking inference and plotting**
   - Limitation: Heatmap generation and model inference execute synchronously on the Flask thread, risking request time-outs and poor responsiveness under load.
   - Solution: Delegate long-running tasks to a background queue (e.g. Celery or RQ) and return immediate acknowledgements, with status polling or WebSocket updates.
3. **Hard-coded process parameters**
   - Limitation: Mesh resolutions, DOE cell mappings, and supported wafer sizes are embedded in code; extending to new geometries or chemistries requires code changes.
   - Solution: Externalize all geometry definitions, DOE templates, and mapping rules into JSON/YAML configuration files, loaded at runtime.
4. **No authentication or authorization**
   - Limitation: The application lacks user login and role-based access control, exposing sensitive process data to any visitor.
   - Solution: Integrate an authentication layer (e.g. Flask-Login or OAuth2/JWT) and enforce role-based permissions to restrict data access and editing.

iii) What is/are the programming language(s) for the software implementation and what platform(s) is it designed for delivery?

- **Programming languages**
  - Backend: Python (Flask framework), leveraging libraries such as Pandas, NumPy, SciPy, Matplotlib, PyTorch, and Joblib
  - Frontend: HTML5, CSS3, and vanilla JavaScript (with Jinja2 templates for server-side rendering)
- **Target platforms**
  - The software is designed as a cross-platform application that can run on standard Windows, Linux, or macOS computers. Users can launch the application directly on their computers without complex installation or configuration. Once started, the application provides an interactive interface that can be accessed through a web browser or directly within the software itself. No additional hardware or special system requirements are needed beyond a typical modern computer.

iv) Is this software? *(You can select multiple if applicable)*

☐SDK ☐Library ☐UI based Application ☐Command line Application ☐Mobile App [✓ ☒Web App] ☐Others

If 'Others', please state: _________

3. COMMERCIALISATION

i) List the following details of associated TD/Patent/Project(s) from which the software IP is originated

TD/Patent # (if any associated TD/Patent)	TD/Patent Title	PSN (e.g. EC-20XX-YY) associated project from which the SWIP is originated	Remarks
TD2025036	AI-Based Optimization for Semiconductor Manufacturing	EC-2024-046	The 2 TDs are from the same project (EC-2024-046 Development of a TCAD-Assisted AI Model for Reducing Process DOE) that the SWIP was developed.
TD2025037	Enhancing measurement integrity in Thin-Film Epitaxial wafers via Anomaly detection and data imputation	EC-2024-046	The 2 TDs are from the same project (EC-2024-046 Development of a TCAD-Assisted AI Model for Reducing Process DOE) that the SWIP was developed.

ii) Is any part of the source code obtained under an **Open-Source** license (e.g. BSD, GPL, MIT, Apache, etc.)?
☐Yes ☐No
If yes, please provide a list of the sources:
(Use of GPL is not recommended for its stringent terms and conditions towards distribution and commercialization)

Name/Title of the open-source software used, source URL and version #	License type (e.g., BSD, MIT, Apache, GPL etc.)	Has the open source used as a library (Y/N)?	Has the source code of the open source modified (Y/N)?
PyTorch	BSD	Y	N
Flask	BSD	Y	N
SciPy	BSD	Y	N
Hyperopt	BSD	Y	N
Scikit-learn	BSD	Y	N

iii) Does the software require any **3rd party / commercial software** (e.g. a 3rd party library?) ☐Yes [✓ No]
(3rd party software is NOT an open source. Example: a shareware/freeware)
If yes, please provide a list of the sources:

Name/Title of the 3rd party software used, source and version #	Type of the 3rd party software (Freeware/ Shareware /commercial)	Is it Free of Cost (Yes/No)?	Remarks

iv) Are there any third-party rights associated with the creation of the software? ☐Yes [✓ No]
List grants or contracts if any, with third parties.

v) What is the TRL (Technology Readiness Level) of the software (**TRL: 1-9**)? 7

[Please refer to this link for more information on TRL: TRL-info.pdf (a-star.edu.sg)

Is the software? [✓ A working prototype] ☐Tested in fully functional end user version?
(if used/tested in user environment, please give details)

4. SUBMITTER/S' PARTICULARS & DECLARATIONS *(Signatures are required.)*	
We* hereby declare to the best of our* knowledge the information provided in this software submission form are true and correct. The uploaded software in gitlab is the complete code, libraries and team has installed the software successfully to run and test as working software.	
Software IP creators/developer(s) Names *(**Please note** that SWIP is the engineering effort to make a reusable software with quality. Thus, the creator(s) should be those involved in realization of the software. The list of the SWIP creators or developers may not be the same as inventors in the associated TD/Patent. Include only those creators for the Software IP.)*	
i) **Principal** SWIP Creator's **Name, Signature & Date** Qian Peisheng **Qian Peisheng** 4 June 2025	Department/Unit/Programme MI, I2R
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>**. Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as involved in coordination, procurement, meetings etc. will not qualify for creator's contribution)* 30%. Peisheng is the main developer of the GUI. He handled front and back-end development, data structure and DevOps (version control, testing, etc).	
Other creators' names and contributions (add more creators if applicable)	
ii) SWIP Creator's Name, Signature & Date Jun Hyunsu Jun Hyunsu , 4 June 2025	Department/Unit/Programme APM, IME
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>**. Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as involved in coordination, procurement, meetings etc. will not qualify for creator's contribution* 30%. Hyunsu designed the overall appearance of GUI (home page, data and model pages), and provided crucial guides on the content and the layout of the GUI.	
iii) SWIP Creator's Name, Signature & Date [illegible] Ashish James , 7 June 2025	Department/Unit/Programme MI, I2R
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>**. Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as involved in coordination, procurement, meetings etc. will not qualify for creator's contribution* 15%. Ashish planned and tracked the development of the GUI, and designed the integration of machine learning algorithms into the GUI. He also implemented the inverse prediction algorithms.	
iv) SWIP Creator's Name, Signature & Date [illegible] Cheng Zhongyao , 4 June 2025	Department/Unit/Programme MI, I2R
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>**. Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as coordination, procurement, meetings etc. will not qualify for creator's contribution* 5%. He provided the training, inference and visualization script using neural networks (LSTM).	

v) SWIP Creator's Name, Signature & Date [illegible] Zhuang Furen , 4 June 2025	Department/Unit/Programme MI, I2R
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>***. *Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as involved in coordination, procurement, meetings etc. will not qualify for creator's contribution* 5%. He provided the training, inference and visualization script using Gaussian Process.	
vi) SWIP Creator's Name, Signature & Date [illegible] Nguyen Xuan Sang , 4 June 2025	Department/Unit/Programme APM, IME
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>***. *Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as involved in coordination, procurement, meetings etc. will not qualify for creator's contribution* 5%. He helped with implementing data preparation, formatting and interpretation.	
vii) SWIP Creator's Name, Signature & Date [illegible] Shiv Kumar , 4 June 2025	Department/Unit/Programme APM, IME
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>***. *Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as coordination, procurement, meetings etc. will not qualify for creator's contribution* 5%. He helped with implementing data preparation, formatting and interpretation.	
viii) SWIP Creator's Name, Signature & Date Navab Singh Navab Singh , 4 June 2025	Department/Unit/Programme APM, IME
Percentage and Aspects of Contribution (***must enter both <u>% and aspects</u>***. *Example of <u>aspects</u>: design, architecting, coding, integrating, validating etc. Non-engineering aspects such as involved in coordination, procurement, meetings etc. will not qualify for creator's contribution* 5%. He provided support for the AI project and GUI development, proposing the integration of both forward prediction (to estimate outcomes from inputs) and reverse prediction (to infer inputs or causes from outcomes).	
Supported by: Li Xiaoli	
Name, Signature of Head of Department/Unit/Programme and Date [illegible] Li Xiaoli, MI, I2R , 20 June 2025	

EVALUATION

SWIP ID/Title	**Reviewed by**
SWIP-/ <SWIP Title>	*Reviewer , *Reviewer

*Consolidate ratings, assessment and approval

Software Acceptance Criteria
[For Submitter Completion: Please fill the relevant columns.]

Category	**Topics**	**Criteria**	**Please tick √** [to be filled up by **submitte**r]			[To be filled up by review committee]	**Remarks**
			Yes	**No**	**NA**	**Rating Scale (1-5)** **5** = Outstanding **4** = Good **3** = Satisfactory **2** = Poor **1** = Unsatisfactory	
Software Usability and Maintainability	Documentation	[†]**Technical guide document** (to prepare using the given template and to be submitted later prior to review) contains release note, installation steps, configuration and how to re-compile & re-build the software. It should spell out the operating computing requirement and environment to run the software, step by step user guide and technical information.	√	■	■		
	Documentation	SDK Manual / API documentation (if applicable) is mandatory for SDK / libraries software release. (source code documentation be generated using tools like Doxygen)			√		Not a SDK
	Documentation	All functions/procedures in source code must have a clear and concise comment blocks with input, output and processing description for easier understanding and maintenance. *Note: Comments with the source codes should focus on function interaction, design approach rather than literally explaining what the code does.*	√		■		
Software Release Readiness	Coding Standard	Comments / code ratio is recommended to be about 40%.	√		■		
	Coding Standard	Software Version is displayed on application GUI (if applicable) correctly during application execution	√				

	System Testing	System Test Cases covering 100% of the features has been created and test results are documented.	√				
	Memory Leak Test for C/C++	Memory Leak Test has been conducted and passed. All defects found are recorded in defect tracking tool.			√		
	†Unit Testing	Unit Testing has been conducted, and unit test results are documented.	√				
	†Code Review (Static Code Analysis)	Static Code Analysis has been performed. Its results have been reviewed.	√				
	†Defect Tracking	Defect tracking (gitlab issue tracker tool) is used to track defects found during software development	√		■		Not in the SWIP submission but in the project Gitlab. I can provide proof of it during demo / review.
Software Quality	Defect Tracking	There are no known open critical and major defects for a release software	√		■		
	Modular Design	Application is partitioned into different standalone libraries with no dependencies on each other	√				
	Modular Design	Each module has well defined APIs	√				
	†Coding Standard	Cross Platform Programming Languages should be used. Preferred Language: C/C++, Java, Python, PHP, JavaScript, R, Objective C. If other language is used, a justification is needed	√		■		

† **Technical guide template,** User guides on Unit testing, code review (aka static code analysis), defect tracking and Coding standard are available in intranet in SWIP site. The artefacts need to be uploaded in gitlab at least 3 working days prior to review.

Evaluation Report
[This section is for reviewer post-review evaluation ONLY.]
[This section will be filed as separate evaluation records and not part of the submit form.]

Date of Review: ____________

Overall Assessment:

Approved to be deposited in software IP bank? Yes/No

Reasons:

Reviewer's Name, Signature:

____________________, ______________________________

Date: ____________