Software Requirements Specification (SRS) & Project Plan

NexDerm - Version 0

Group 11

COMPSCI 4ZP6

Dr. Mehdi Moradi

October 10, 2025

Divye Goswami	goswad3@mcmaster.ca	400362533
Tanvir Singh	singht48@mcmaster.ca	400378756
Hetanshu Pandya	pandyh6@mcmaster.ca	400371731
Ishaan Jamwal	jamwali@mcmaster.ca	400389636
Harkunwar Singh	singh42@mcmaster.ca	400380573

1. Title

NexDerm: AI/ML-Powered Mobile Application for Skin Condition/Disease Classification and Dermatology Support Based on Captured/Uploaded Images.

2. Version

Version 0 - Initial submission of the Software Requirements Specification and Project Plan

3. Personnel of the Project & Roles

Member	Emails	Roles
Divye Goswami	goswad3@mcmaster.ca	Frontend Developer & AI/ML Engineer
Tanvir Singh	singht48@mcmaster.ca	Backend Developer & AI/ML Engineer
Hetanshu Pandya	pandyh6@mcmaster.ca	Frontend Developer & AI/ML Engineer
Ishaan Jamwal	jamwali@mcmaster.ca	Backend Developer & AI/ML Engineer
Harkunwar Singh	singh42@mcmaster.ca	Backend Developer & AI/ML Engineer

4. Abbreviations, Notations & Definitions

- **ML** -Machine Learning: Algorithms and models that enable predictive capabilities from datasets.
- **AI** -Artificial Intelligence: Broader category encompassing ML and decision-making systems.
- **API** -Application Programming Interface: Defines communication between software layers (backend ↔ frontend).
- **DB** -Database: Persistent storage for structured data such as user profiles and logs.
- **UI/UX** -User Interface/User Experience: Visual and interaction design of the mobile application.
- **F1 Macro –** F1 Macro is Harmonic means of precision and recall, appropriate for imbalanced datasets.
- **AUC** -Area Under Curve: Commonly AUC-ROC, indicates classifier separability across classes.
- **TFLite** - TensorFlow Lite: Lightweight ML framework optimized for mobile inference.
- **Core ML** - Apple's ML framework for iOS inference.
- **CV** - Cross Validation is a statistical method for robust model evaluation.
- **CRUD** - Create, Read, Update, and Delete, which are the four basic functions of persistent storage and are essential for managing data in applications and databases.
- **ETL** - Extract, Transform, Load - Data Preprocessing for data analytics and machine learning projects.

- **WCAG -** Web Content Accessibility Guidelines, are international standards developed by the World Wide Web Consortium (W3C) to make digital content, such as websites and applications, accessible to people with a wide range of disabilities, including visual, auditory, physical, cognitive, and neurological disabilities.
- **PIPEDA** - Personal Information Protection and Electronic Documents Act is a Canadian federal law that governs how private-sector organizations collect, use, and disclose personal information during commercial activities.

5. Table of Contents & Contributions

Table of content at the start of the SRS document.

Group Member	Contributions (Sections)
Divye Goswami	1, 2, 10, 13, 18
Tanvir Singh	6, 9, 17, 18
Hetanshu Pandya	3, 7, 8, 15, 18
Ishaan Jamwal	4, 12, 16, 18
Harkunwar Singh	5, 11, 14, 18

6. Purpose of the Project

6.1 Background

Skin diseases are one of the most prevalent health problems in the world. However, the availability of dermatologists is often limited by either extensive wait times or geographic locations that may be far away. New developments in AI and computer vision have provided the possibility of automated classification of skin images, but the majority of already existing tools are either proprietary, expensive or require a specific infrastructure to use. Our goal is to establish an accessible, informative, and private AI-assisted dermatology application that provides real-time image classification and then later provides support locating a dermatologist through a mobile platform.

6.2 Problem Statement

People often consult unreliable online sources for self-evaluation of skin conditions due to delays in care. Current AI solutions are driven by commercial, closed source products that are not amenable to academic or open-source development processes. There is no transparent and low-cost option for individuals to responsibly assess and analyze skin images while being mindful of privacy and promoting consultation with a physician.

6.3 Problem Objectives

The project's primary goals include the development of a mobile-first application that classifies

images of skin-conditions using an external inference API, provides users with information about dermatologists located nearby through Google Maps integration, and maintain data privacy through user-consent elements. NexDerm also aims to demonstrate ethical and explainable use of AI/ML within healthcare by providing a system that carries a full end-to-end system designed for use and evaluation by researchers.

6.4 Proposed Solution

NexDerm will deliver a safe mobile application that allows users to either take or upload skin images, receive classification results along with a confidence score, and find local dermatologists. The AI model itself, which has been based on EfficientNet-B2, will be hosted on a secure external server for performance reasons, owing to the expected heavier loads from multiple images. Users will have the option to consent to the use of their anonymized images to improve the accuracy of the model. The focus for this project will continue to be on accessibility, reliability, and responsible AI use, rather than any clinically diagnostic use.
The NexDerm project will function as a proof-of-concept project for the responsible use of medical AI, focusing on accessibility, transparency, and ethical AI data responsibilities, while still fulfilling the functional and non-functional architecture templates discussed in a later section.

7. Clients & Stakeholders

7.1 Clients (Primary Users):

- The general public concerned about skin conditions.
- Patients in remote or underserved regions with limited access to dermatologists.
- Medical students and primary care practitioners for learning and triage support.

7.2 Stakeholders (Secondary Participants):

- Dr. Mehdi Moradi
- Amirhossein Sabour
- Sahib Khokhar
- Dermatologists and clinics - potential referral recipients and validation partners.
- Dataset providers (Kaggle/ISIC) - indirect stakeholders providing training data.
- Regulatory authorities (PHIPA/PIPEDA) - data privacy compliance (if launched to the public world).
- Technology providers - Apple (Core ML), Google (TFLite, Maps/Places API), Supabase.

8. Project Constraints

Several high-level design decisions and constraints, whether determined internally or externally, will influence the work regarding the Skin Disease Detection System. These constraints create limits for the system and affect related decisions about a system's architecture, and impact on related decisions around technology and functionality.

8.1 Platform Constraint

The system will be utilized as a mobile application with a user interface. Because the aim is to provide an engaging and user-friendly solution that end-users can access, a backend or research prototype has been removed from consideration.

8.2 Model Development Constraint

A pretrained deep learning model will be employed to create the machine learning aspect of the system and a selected dataset of dermatological images will be used to fine-tune the model. Due to constraints of project time, compute resources and training data, any option to train a model from scratch has been discounted. This constraint ensures that practical, efficient, and diagnostic accuracy is maintained.

8.3 Functional Scope Constraint

- Users will have the option to upload pictures of their skin to the system, which will automatically evaluate the images for diagnostic purposes.
- To determine the category of the disease in the supported range, the classifier has to first demonstrate that the user has a healthy or diseased skin.

- The system will also provide context-aware suggestions for dermatologists in the user's area depending on the data and availability of external APIs.

8.4 Data Management Constraint

- The diagnostic results of the users will be saved in a protected database associated with the user.
- The sensitive data must have encryption and follow upheld privacy and security practices.
- The data retention policy will follow ethical standards for medical apps and conform to only the data necessary to operate and improve the system's capabilities.

8.5 Applicability and Limitation Constraint

- The system has been specifically created to act as an ancillary, diagnostic tool and not as a substitute for professional clinical judgment. Users will be cautioned that results are susceptible to chance and should not be regarded as merely conclusive
- The variability, representation, and quality of the training dataset will all have an impact on predictions. There may be reduced predictive accuracy in cases involving rare conditions or images containing low quality.
- Depending on the availability of the datasets, or third-party service integrations, the dermatologist's recommendation function may only operate in limited geographic settings.
- If a user does not enable camera permissions using ordinary device functions, the functionality of the app may be limited.

8.6 Technological Constraint

In order to support scalability, maintainability, and reliability, the project will use open-source and widely supported technologies:

- Frontend: TypeScript with React Native and Expo for cross-platform development with React Navigation for routing and Redux for state management.
- Backend: Python containerized with Docker with FastAPI to provide RESTful APIs for reliable deployment.
- Database: PostgreSQL utilizing Supabase's integrated monitoring and authentication capabilities.
- Machine Learning: GPU acceleration on the CAS server with PyTorch and Tensorflow/Keras frameworks using EfficientNet-B2 as our main architecture.
- MLOps: Docker for reproducibility, DVC for dataset/model versioning, GitHub Actions for CI/CD.
- Testing and monitoring using Jest (frontend), Pytest (backend), Supabase logs, API health checks, and W&B dashboards for machine learning monitoring.

8.7 Schedule Constraint

The system is expected to have functioning proof of concept by November 20th, 2025, and a fully functional application before 22nd March 2026.

9. Functional Requirements

9.1 - Priority 0 (Minimum Viable Product)

- User must be able to 'Continue as Guest' (i.e. upload and get results without logging in).
- Users must be able to upload a skin image through the application of UI.
- An API to allow for the submission of uploaded images to the backend for processing.
- The backend must process and validate uploaded images.
    - Data Needed: Image uploaded by user
- A trained deep learning model must classify the uploaded image into one of the supported skin disease categories or 'healthy'.
    - Data Needed: Training data for image classification model sourced from Kaggle.
- The backend must provide an API endpoint for classification of requests.
- The application must display the classification result to the user in a clear and accessible format.
    - Data Needed: Classification result from model
- If the prediction of confidence is below a threshold, the app must notify the user that results are uncertain.
    - Users must be able to interact with the application via a mobile interface.
- Software must include basic error handling (invalid input, server errors).
- Documentation of backend APIs.
- Unit testing for model inference API.

9.2 - Priority 1 (Next Steps)

- Users must be able to capture images directly using their device camera in addition to uploading.
- Users must be able to create a new account and login using an email and a password.
- An API to send user registration information to the backend.
- A database with CRUD functionality and API endpoints for user authentication and for saving user history.
- If a potential condition is detected, the system must suggest nearby dermatologists using geolocation data.
- The backend must integrate with a location-based API (e.g., Google Maps Places API) to suggest nearby dermatologists.
    - Data Needed: API response from Places API
- The app must store a history of uploaded images and results for authenticated users.
- Intuitive UI with a mobile-friendly design for displaying predictions, dermatologist suggestions, and past results.
- The backend must connect to an external API for a text report generation model.
- The application must be able to generate a report (pdf) with the results of the visual diagnosis.
    - Data Needed: Classification result from model, API response from textual report generation model

- The user should be able to view their history of diagnosis and compare results with past diagnoses.

9.3 - Priority 2 (Non-Critical Features)

- Highlight areas of the image that influenced the model's decision (e.g., heatmaps/Grad-CAM/DinoV2).
    - Data Needed: Classification result (on train and CV datasets)
- Provide informational content about each skin condition (non-diagnostic guidance).
    - Data Needed: Classification result
- Users must be able to download a PDF summary of the results.
    - Data Needed: Classification result from model, API response from textual report generation model
- The application should have two separate classes of users ('Patients' and 'Doctors')
- Users may optionally share results securely with healthcare providers.
- The backend must connect to an external API for a text report generation model.

9.4 - Priority 3 (Future Application Features)

- The application should support basic accessibility features (based on WCAG standards).
- Allow users (with consent) to contribute their images to improve the dataset for future model retraining.

10. Data & Metrics

This section describes the data sources, handling methods, and performance metrics used for the functional requirements identified in Section 9. Since NexDerm performs inference through an API, we emphasize not forgetting the data ethics, validation, and any realistic measurable performance goals.

10.1 Dataset

The dataset (https://www.kaggle.com/datasets/ahmedxc4/skin-ds/data) includes approximately 10 GB of dermatological images that provided labeled samples for 14 classes, specifically, Actinic Keratoses, Basal Cell Carcinoma, Benign Keratosis-like Lesions, Chickenpox, Cowpox, Dermatofibroma, Hand-Foot-Mouth Disease (HFMD), Healthy, Measles, Melanocytic Nevi, Melanoma, Monkeypox, Squamous Cell Carcinoma, and Vascular Lesions.

A separate file named label_map.json provides a one-to-one mapping between the numeric class indices of predictions and the human friendly names for those classes. This way, the model, the backend (FastAPI), and the frontend (React Native) always have the same understanding of predictions and correct labels (in the event there is a difference in the order of classes).

Feature	Data Used / Source	Purpose
Image Upload & Classification	Skin-DS + optional user-consented data	Train and evaluate EfficientNet-B2 classifier.
Dermatologist Locator	Google Places API	Retrieve nearby dermatologists.
Report Generation	Classification result + metadata	Create downloadable reports for users.

10.2 Data Source and Acquisition Plan

- **Source of Data**: The Skin-DS dataset is publicly accessible on Kaggle.
- **Training Facility**: The training will take place on McMaster's GPU servers.
- **User Data and Privacy Practices**: Images uploaded are processed only once and will then be deleted unless the user consents to allow the team to anonymously reuse the images to develop and refine the model further.
- **Ethics and Storage**: Consent flags are saved in a PostgreSQL (Supabase) database; no images will be kept or stored if users do not consent.

10.3 Performance Metrics, Relevance, and Goals

Metric	Target	Purpose / Relevance
Top-1 Accuracy	≥ 60 %	Overall, correctness across all classes.
F1-Macro	≥ 0.50	Balances precision & recall for imbalanced data.
AUC (ROC Curve)	≥ 0.70	Evaluates threshold-independent discrimination.
Confusion Matrix	NA	Highlights class-wise misclassifications.
Functional Success Rate	100 %	Confirms complete backend-to-frontend workflow.
API Latency (Mean)	≤ 3 s	Measures responsiveness of external API.

Validation Plan:
Metrics are computed on the held-out test set; ROC curves and AUC scores are plotted per class. Latency and functional checks ensure smooth operation, and consent logs are verified for compliance.

Acceptance Criteria:
Model meets ≥ 60 % accuracy, ≥ 0.50 F1-Macro, and ≥ 0.70 AUC with all core flows functional and responses under 3 s.
Consent system must prevent any data retention without permission.

Future Plan:
Metrics will be re-evaluated in Milestone 4, and retraining anonymized opt-in data will be pursued if targets are not met.

11. Non-Functional Requirements

11.1 Look and Feel Requirements

- The mobile application will use a clean, professional-looking design consistent with healthcare-related design expectations.
- Colors, typography, and layouts should appear consistently around the application to promote readability and a presence of user trust.
- The display of prediction results should be organized, coherent, and styled to be polite without being alarming, but still be able to direct the user to a more significant finding.
- The design should be responsive and usable through mobile devices on various screen sizes.

11.2 Usability and Humanity Requirements

- The application must be intuitive and easy to navigate, requiring no technical or medical experience.
- All error messages (e.g., uploading an image in an invalid format, cannot connect to server) must be clear, actionable and non-technical.
- The application must accommodate users' varying visual abilities, such as text scaling or high contrast themes.
- The app must remind the user through disclaimers that the information provided is not a medical diagnosis and they should speak to a physician.
- The application should provide a consistent user experience for both registered and guest users.

11.3 Performance and Speed Requirements

- Images that are ≤5MB must be uploaded and classified in a few seconds, assuming normal network conditions.
- The database must provide CRUD functionality (e.g., creating an account or a user session, authenticating a user, uploading images, and retrieving image upload history or classification history), with an average response of ≤ 10 seconds for CRUD operations, assuming normal operating conditions.
- The backend must be able to support at least ten concurrent requests to classify images without significant performance degradation.

- The model that has been developed must achieve a minimum accuracy of 60%, with the goal of classification on a validation data set, before being considered reliable enough for deployment.

11.4 Security and Privacy Requirements

- All communication between the frontend and backend must utilize HTTPS/TLS encryption.
- User credentials must be stored securely, using salted and hashed passwords.
- Uploaded skin images must be handled in a way that is in compliance with privacy standards. The skin images must be encrypted in storage and transmission.
- Images uploaded by guest users must only be used for real-time inferences and must not be stored longer.
- There must be a way for the users to delete their account and associated data if they request it.
- Regular backups must be performed of non-sensitive data related to the system (e.g., logs, metadata) to make sure it is available, as well as nature to enable recovery in case the system fails.
- While the GitHub repository may remain open and accessible for transparency and collaboration, administrative actions of extreme importance (e.g., merging pull requests, changing core functionality) must be limited to "trusted" stakeholders (e.g., professors, TAs, developers that have received explicit authorizations) to ensure accountability and avoid unauthorized access.
- Regular backups of source code and system data must occur to ensure recovery if the system fails or is the target of a security breach. The team is encouraged to regularly commit their codes to the GitHub repository, so in the event of failure of their computing device (e.g., laptop crash), their latest work remains accessible from the public storage unit.

11.5 Legal Requirements

- The application must display a disclaimer stating that it is not a replacement for professional medical advice, and that the user should consult a dermatologist for proper diagnosis.
- The application must follow app store guidelines or policies.
- Any third-party integration (like Google Maps API) also must follow their terms of service.
- The project must use an open-source MIT license or a license that is agreed upon by the parties.
- This is a student project/capstone project, and the application is not for actual clinical use; however, if it is used in a healthcare environment, the application must also comply with relevant data protection regulations (PIPEDA in Canada, PHIPA in Ontario, GDPR in EU, and HIPAA in the US for example).

12. Risks and Issues

12.1 Open Problems

- **Model Effectiveness and Misclassification:** The classifier sometimes misclassifies some conditions or misses detection of conditions altogether. This remains a current, ongoing issue, through the incapacity of today's deep learning models in the medical sphere.
- **Data Limitations:** The Kaggle dataset may not accurately represent all skin tones, age categories, or rare dermatological conditions. This ultimately leads to biased outputs and reduced generalizability.
- **Moral Responsibility:** The system could be misused as a diagnostic tool, notwithstanding that its intended use is a support application. Prevention of this misuse would be done via disclaimers and clear communication to all users.
- **Computational Constraints:** Training and inference of this AI application would require the use of significant GPU/CPU resources. If you only have limited infrastructure, this will impact performance.

12.2 Predicted Risks

- **Data Privacy and Security Threats:** There is a chance that users will not want to upload sensitive medical images. Any vulnerabilities related to storage, transit, or access could impact user trust.
- **Regulatory and Legal Threats:** Given that the project is based on health data, frameworks such as GDPR or HIPAA may be an issue. Even for a Capstone project, if we ignore these considerations, we may still face reputational risk.
- **User Trust and Adoption Threats:** Users, regardless of technical accuracy, might not trust AI generated predictions without the input of a medical professional. Users will be unlikely to adopt the tool without the support of explicit disclaimers, endorsement, or even optional validation by a dermatologist.
- **Maintenance Threats:** Dermatology evolves as new conditions, treatments, and new classifications emerge. If the system is not maintained after the Capstone project, accuracy will continue to decline as will relevance.
- **Risks Involving Team and Timelines:** The initiative depends on experts in machine learning, mobile/web development, cloud hosting, and security. If we are missing any one of these skills during a critical phase, or if academic responsibilities take up unforeseen time commitments, this could pose risks of delays for milestones or affect the number of features.
- **Risks Involving Image Quality or Variance in Capturing Images:** The quality of the uploaded photos is highly correlated with effective predictions. Images that are poorly lit, low resolution, have motion blur, or are taken from poor angles will negatively impact model prediction. Variance in the way in which users choose to capture images (e.g. different phone cameras, distance from skin, lack of scale in the photo, etc.) can also

introduce noise or inconsistency in the classifier, making classification unreliable. This could require creating a standard photo capturing protocol or simply built-in capture assistance within the app.

12.3 General Issues

- **Collaboration with External Services:** To recommend dermatologists in the area, we will need to integrate with some kind of location service, and with a third-party API such as Google Maps. The problem may arise when there are limits on the API we use, the cost, or availability.
- **Scalability:** This is not an issue for the Capstone project, but when the application is deployed in the real world, it may pose an issue at scale (for thousands of users). This would also involve some substantial back-end infrastructure that likely would not be available.
- **Bias and Fairness Issues:** There is the potential for the application to perform differently for different demographic groups, which would raise larger ethical considerations. In this case, simply an ethical reason, we would need additional datasets beyond just the Kaggle dataset.

13. Team Meeting & Communication Plan

Our team follows a structured but flexible communication and collaboration process to ensure consistent progress throughout the project.

13.1 Meeting Schedule and Format

The group will gather in-person a minimum of three times a week (Monday, Wednesday, and Friday), for a minimum of two hours for each meeting, in allocated library rooms. If there are any conflicts (e.g., exams, other events), we will move our meetings online via Teams or Discord, or to alternative times. At the end of each meeting, we will collaboratively create an agenda for the following meeting, to discuss updates on work completed (each member's work), blockers (if any), and next steps. All contributions are equal. There is no fixed leader.

13.2 Communication & Collaboration Tools

- Task tracking is done using Jira, while Confluence is used to store meeting notes.
- Version control and code reviews are done using GitHub.
- Formal communication with instructors and TAs is done via email and Teams, while Discord is used for internal team communication.

13.3 Project Tracking and Monitoring

Tasks will be parsed into short, manageable tasks that are represented in a Kanban-like workflow in Jira. Milestones and deadlines are being monitored with the tools of Jira, Confluence, and Google Calendar. When the team meets weekly to sync, task status will be reviewed for commitments and accountability.

13.4 Conflict and Blocker Resolution

- When issues or blockers arise, they are recorded as a Jira ticket issue, discussed in meetings, and assigned to members who can resolve them.
- The team interacts openly and respectfully to avoid conflicts and delays.
- If something cannot be resolved, it will be escalated to the course instructor or TA.

14. Team Member Roles

14.1 Functional Requirements

Functional Requirement	**Primary Developer**	**Secondary Developer**
Guest Upload & Mobile UI	Divye Goswami	Hetanshu Pandya
Camera Capture Integration	Hetanshu Pandya	Divye Goswami
Image Ingestion & Validation	Tanvir Singh	Harkunwar Singh

Inference API (AI/ML)	All Members (Divye, Hetanshu, Tanvir, Ishaan, Harkunwar)	All Members
ML Model & Data (AI/ML)	All Members (Divye, Hetanshu, Tanvir, Ishaan, Harkunwar)	All Members
Results UX & Accessibility	Hetanshu Pandya	Divye Goswami
Confidence & Uncertainty (AI/ML)	All Members (Divye, Hetanshu, Tanvir, Ishaan, Harkunwar)	All Members
Auth & Roles (Patients/Doctors)	Ishaan Jamwal	Tanvir Singh
History & Comparison	Ishaan Jamwal	Harkunwar Singh
Dermatologist Suggestions (Google Maps API)	Harkunwar Singh	Tanvir Singh
Report Generation (PDF & External Text API)	Harkunwar Singh	Ishaan Jamwal
Secure Sharing of Results	Ishaan Jamwal	Divye Goswami
Explainability & Education (AI/ML)	All Members (Divye, Hetanshu, Tanvir, Ishaan, Harkunwar)	All Members
Reliability & Error Handling	Tanvir Singh	Ishaan Jamwal
Documentation & Testing	All Members (Divye, Hetanshu, Tanvir, Ishaan, Harkunwar)	All Members

14.2 Non-Functional Requirements

Non-Functional Requirements	**Primary Developer**	**Secondary Developer**
Look and Feel Requirements	Divye Goswami	Hetanshu Pandya
Usability and Humanity	Hetanshu Pandya	Divye Goswami
Performance and speed requirements	Tanvir Singh	Ishaan Jamwal
Security and Privacy	Ishaan Jamwal	Harkunwar Singh

Legal	Harkunwar Singh	Tanvir Singh

We as a group have decided that the Coordinator / Program Manager would be Ishaan Jamwal(jamwali@mcmaster.ca).
Ishaan will supervise the development activities, ensures collaboration between frontend and backend teams, monitors schedule deadlines and deadlines, and will track the progress of project milestones. Ishaan is also responsible for the communication and integration across all the modules ensuring consistency and quality of the project.

15. Workflow Plan

15.1 GitHub management

The team plans to leverage GitHub for the application's version control and associated project artifacts. Everyone engaged with the application, including students and the supervising professor and teaching assistants, have been given access to the repository for the purpose of transparency and collaboration. The team will follow a branch-based workflow, generating pull requests for desired changes in code. All pull requests will be reviewed by multiple reviewers to ensure adherence to coding standards, maintainability, and quality assurance before merging the changes to the main branch. In addition, the team will utilize Jira for issue tracking and project management. Issues will be created, assigned and managed within Jira, which will link to the appropriate GitHub branch to clarify the traceability of requirements, code changes, and the progress on project tasks.

15.2 Development methodology

The team will be working in an agile development methodology, in particular, using Kanban. A Kanban board in Jira will be maintained to visualize workflow, support prioritizing tasks, and track project progress in real-time. This approach supports continuous delivery, increased flexibility to manage changing requirements, and boosts team productivity.

15.3 Data storage

Given that our dataset is structured, it makes sense to store it in a SQL database like PostgreSQL for efficient querying, reliability, and scalability of structured records. PostgreSQL offers sophisticated indexing and transaction management potential, making it a good fit for managing training data while also preserving the integrity of the data.

15.4 Training Models

We will use the McMaster University servers for compute-intensive tasks such as model training and fine-tuning. These servers include the hardware resources needed (e.g. GPUs/CPUs, large memory, high-speed storage) to train machine learning models at scale. By training in these performance-driven systems, we will have faster execution times, and an opportunity to experiment with larger models and datasets without limitations from local hardware.

15.5 Tools/Methods for success

This pairing of -PostgreSQL for structured data management, and McMaster servers for heavy computation- meets the performance requirements outlined in the SRS, in that it meets data handling needs in an efficient manner and trains ML models in a timely manner.

16. Proof of Concept Demonstration Plan

In this proof of concept demonstration, we will show the primary functionality of the NexDerm application, demonstrating the initial end-to-end user experience. This will include the upload and analysis of a sample skin image through the application interface and display of the classification output generated by the system. The demonstration will use a limited initial version of the deep learning model allowing for initial functional flow testing, and will highlight the interaction of the front-end, back-end, and model components with limited handling of edge cases and validations. Testing outputs, example previews, or initial API responses may also be included to demonstrate the initial integration of elements of the system overall.

17. Technology

17.1 Languages and Libraries

17.1.1 Frontend

We will use **TypeScript** with **React Native** as the front-end framework, supported by **Expo** for mobile application development and **React Navigation** for routing. **Redux** will be used for state management. The development environment will be **VS Code**. Unit testing will be implemented using **Jest**. Here, unit testing would improve reliability, support early bug detection, and ensure each module works independently. Additionally, **Google Maps Platform API** will be used to embed a map interface.

17.1.2 Backend

We will use **Python** and **FastAPI** to develop the backend and REST API endpoints to support authentication, image upload, classification requests, and dermatologist suggestions based on user location. The backend will also connect to the **Google Maps Platform API** to support location-based dermatologist search. Modules and Packages will be containerized using **Docker** for portability and consistent deployment across environments. We will also perform unit testing with **Pytest** to improve reliability, support early bug detection and ensure each module works well independently.

17.1.3 Database

We will set up a **PostgreSQL** database hosted on **Supabase** to store user login data, logs, and the history of uploaded images and classification results. Supabase will also provide authentication support.

17.2 AI/ML Libraries & GPU Usage

We will use **PyTorch** and **TensorFlow/Keras** as the core machine learning frameworks along with **scikit-learn**, **Pandas** and **NumPy** for data preprocessing and ETL. The deep learning pipeline will be built using **Torchvision** with **EfficientNet-B2** as the primary architecture for training. Model training will require **GPU acceleration**, for which we plan to use **CAS's Ada** server. Additionally, we might be using **Grad-CAM** and **DINOv2** for generating heatmaps in the interest of model optimization.

17.3 MLOps & Infrastructure

Continuous integration and deployment (CI/CD) will be managed with **GitHub Actions** which helps in automating testing and deployments. **DVC (Data Version Control)** will be used to manage datasets for model training and testing, and version control for experiments.

18. Project Scheduling (Gantt Chart Overview)