SOFTWARE DESIGN SPECIFICATION

ConnectTree Application

Version 1.0

Prepared by: Jordan Kelley
Buchard Joseph
David Martinez

October 8, 2024

Contents

1 System Description 3

2 Software Architecture Overview 4

2.1 ULM Class Diagram . . . 5

3 Description of Classes, Attributes, and Operations 6

3.1 User . . . 6

3.2 Connections . . . 6

3.3 Settings . . . 7

3.4 ProfileInfo . . . 7

3.5 Personal . . . 7

3.6 Company . . . 8

3.7 Professional . . . 8

3.8 Direct Messaging . . . 8

3.9 GUI . . . 9

3.10 Groupchat . . . 9

3.11 Bubble Map . . . 9

3.12 Chat . . . 9

4 Development Plan and Timeline 11

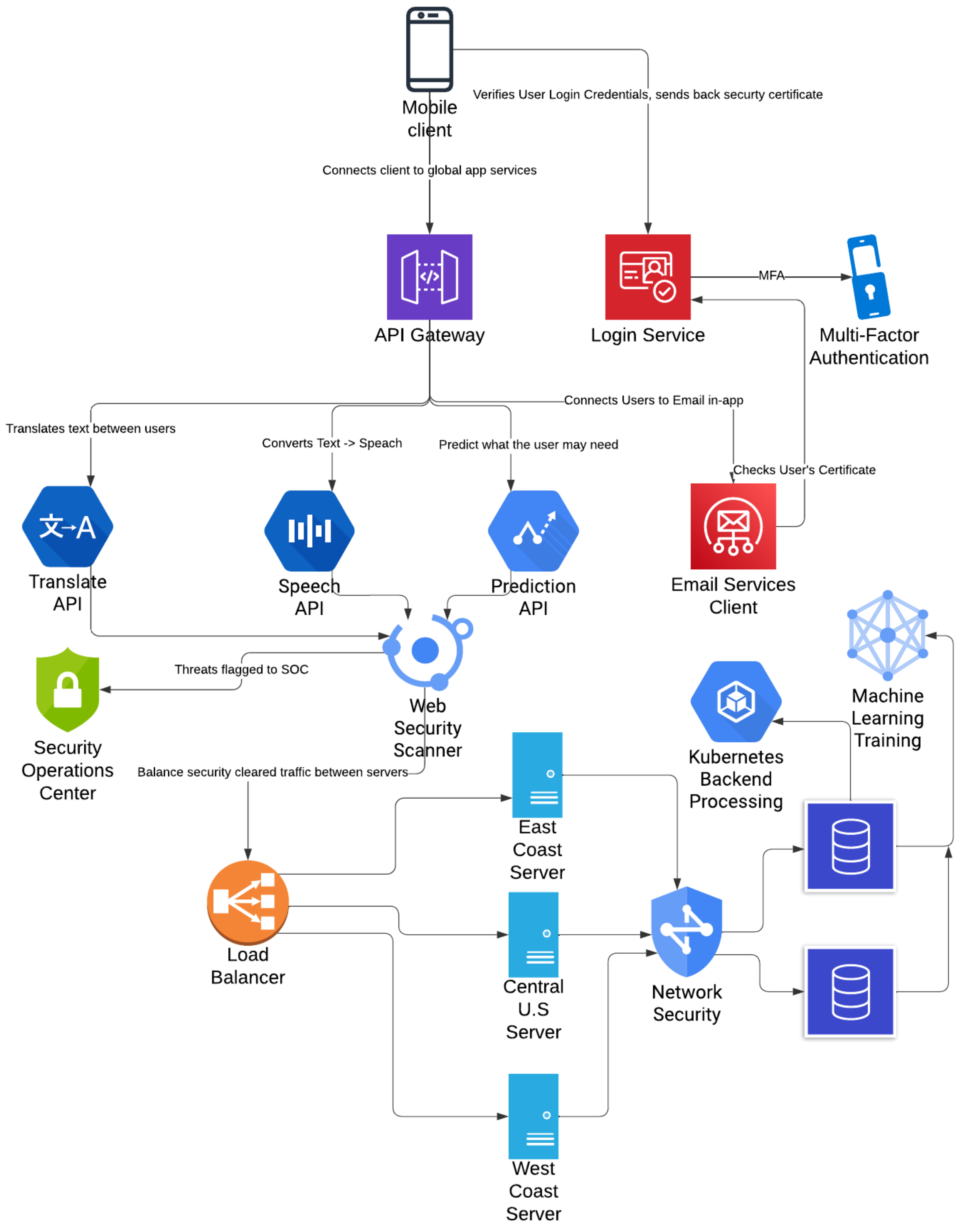
1 System Description

Purpose

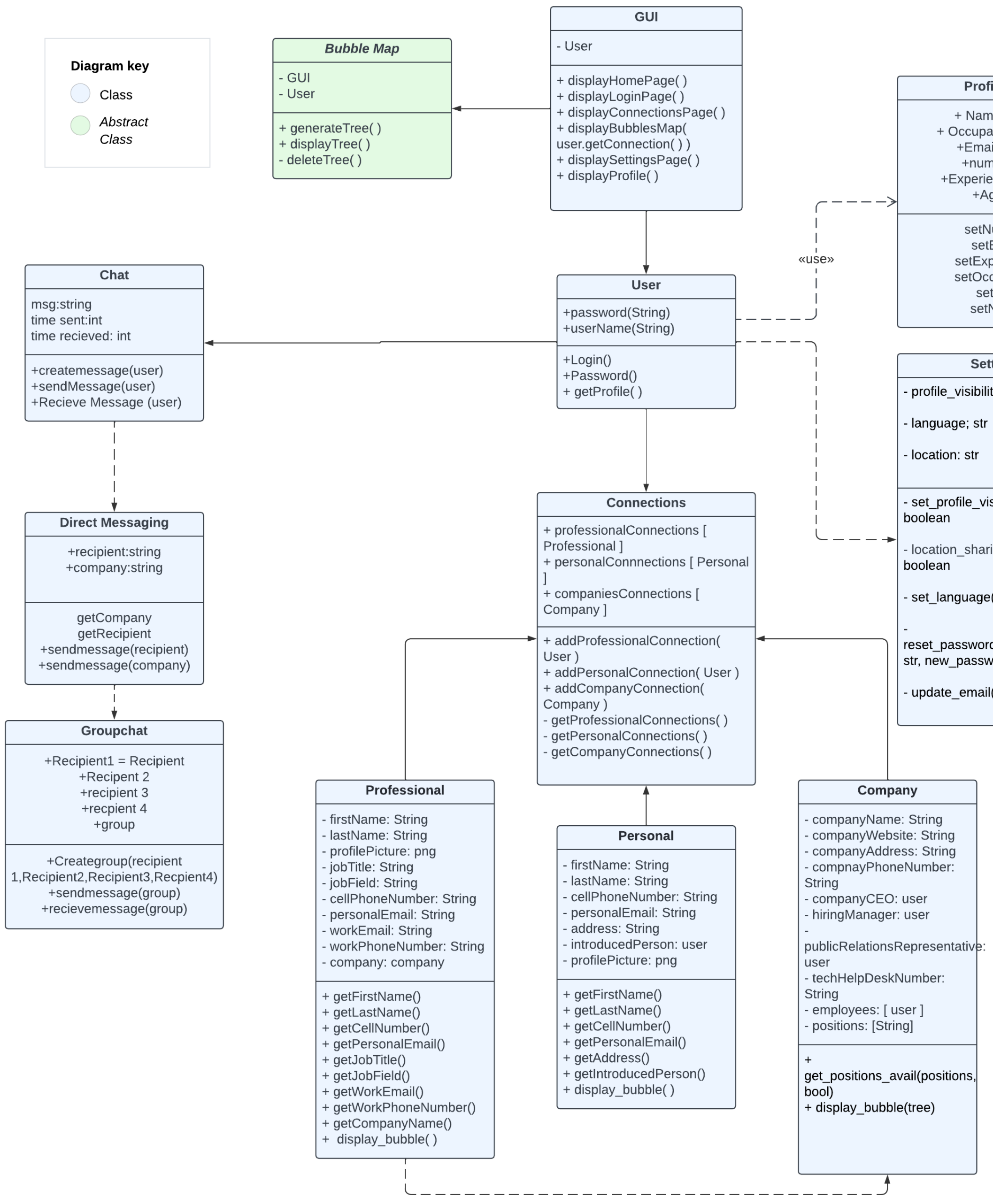
Target Audience

Definitions, acronyms, and abbreviations

2 Software Architecture Overview


Mobile client
Verifies User Login Credentials, sends back securty certificate
Connects client to global app services
API Gateway
Login Service
MFA
Multi-Factor Authentication
Connects Users to Email in-app
Translates text between users
Converts Text -> Speach
Predict what the user may need
Checks User's Certificate
Translate API
Speech API
Prediction API
Email Services Client
Machine Learning Training
Threats flagged to SOC
Web Security Scanner
Security Operations Center
Kubernetes Backend Processing
Balance security cleared traffic between servers
East Coast Server
Load Balancer
Central U.S Server
Network Security
West Coast Server

2.1 ULM Class Diagram



3 Description of Classes, Attributes, and Operations

3.1 User

This class represents a system user with personal login capabilities.

Attributes

- **password (String)**: The user's password for login.
- **userName (String)**: The username for login.

Operations

- **Login()**: Authenticates the user.
- **Password()**: Verifies a password.
- **getProfile()**: Retrieves the user's profile information.

3.2 Connections

Handles the different types of connections a user can have.

Attributes

- Arrays of different connection types like professional, personal, and companies, suggesting each user can have multiple connections of each type.

Operations

- **addProfessionalConnection(User)**: Adds a professional connection.
- **addPersonalConnection(User)**: Adds a personal connection.
- **addCompanyConnection(Company)**: Adds a connection to a company.
- **getProfessionalConnections()**: Returns a list of professional connections.
- **getPersonalConnections()**: Returns a list of personal connections.
- **getCompanyConnections()**: Returns a list of company connections.

3.3 Settings

Manages user settings related to privacy and preferences.

Attributes

- **profile_visibility (bool)**: Boolean indicating if the profile is visible to others.
- **language (str)**: Preferred language setting.
- **location (str)**: User's location.

Operations

- **set_profile_visibility(user)**: Sets the visibility of the user's profile.
- **location_sharing(user)**: Shares or hides the user's location.
- **set_language(language)**: Sets the user's preferred language.
- **reset_password(old_password, new_password)**: Resets the user's password.
- **update_email(new_email)**: Updates the user's email.

3.4 ProfileInfo

Contains detailed personal information about the user.

Attributes

- **Name, Occupation, Email, Number, Experience, Age**: Basic profile information.

Operations

- Various setters (e.g., **setNumber**, **setEmail**, etc.) to update profile information.

3.5 Personal

Details a personal connection.

Attributes

- **firstName**, **lastName**, **cellPhoneNumber**, **personalEmail**, **address**, **introduced-Person**, **profilePicture**: Personal details of the connection.

Operations

- Getters for each attribute.
- **display_bubble()**: Method to display user details visually.

3.6 Company

Represents a company in the user's network.

Attributes

- Company-specific details such as **name**, **website**, **address**, **phone number**, **CEO**, and other relevant personnel.

Operations

- **get_positions_avail(positions, bool)**: Shows available positions within the company.
- **display_bubble(tree)**: Displays information in a visual format.

3.7 Professional

Details a professional connection.

Attributes

- Professional and contact details similar to those found in Personal.

Operations

- Getters for professional details.
- A method to display these details visually.

3.8 Direct Messaging

Facilitates messaging between users.

Attributes

- **recipient**, **company**: Identifiers for message recipients.

Operations

- **getCompany**, **getRecipient**: Retrieve details about the message recipient.
- **sendmessage(recipient/company)**: Send messages to either a company or an individual.

3.9 GUI

Manages the graphical user interface components.

Operations

- Methods to display different user interface pages like home, login, connections, settings, and profile pages.

3.10 Groupchat

Manages messaging within a group.

Attributes

- Recipients' list and a group identifier.

Operations

- **Creategroup**: Creates a new group chat.
- **sendmessage**, **recievemessage**: For sending and receiving messages in a group.

3.11 Bubble Map

Visual representation of connections or data.

Operations

- **generateTree**, **displayTree**, **deleteTree**: Manage visual representation of hierarchical data.

3.12 Chat

Manages chat messages.

Attributes

- **msg**, **time sent**, **time received**: Stores message data and timestamps.

Operations

- **createmessage**, **sendMessage**, **RecieveMessage**: Manage sending and receiving messages.

4 Development Plan and Timeline

Stage	Tasks	Assigned to	Duration (Weeks)
1. Requirements Analysis	Define app objectives, user stories, and connection flow (user profiles, professional, personal connections)	Developer 1	1
	Gather and document detailed functional and non-functional requirements for ConnectTree	Developer 2	1
	Identify constraints, platform dependencies, security considerations for connections	Developer 3	1
2. System Design	Create UI/UX mockups for ConnectTree, including connection bubbles, user interfaces	Developer 1	2
	Define software architecture, class diagrams, and data structures for tree connections	Developer 2	2
	Define API contracts for managing connections and messaging, database schema design	Developer 3	2
3. Implementation	Develop front-end (Swift UI), including user interfaces and bubble map navigation for connections	Developer 1	4
	Implement back-end services (Swift, iOS SDK) for handling professional, personal, and company connections	Developer 2	4
	Set up database integration, user authentication, and real-time messaging services	Developer 3	4
4. Integration and Testing	Unit testing of UI components and navigation	Developer 1	2
	Integration testing between front-end, back-end services, and database	Developer 2	2
	System testing, ensure compliance with Apple guidelines and security tests for connections and messaging	Developer 3	2
5. Deployment	Prepare ConnectTree for submission to Apple App Store	Developer 1	1
	Finalize and review all technical documentation and user manual	Developer 2	1
	Deploy app to TestFlight and collect feedback from beta testers	Developer 3	1
6. Maintenance	Monitor app performance, fix bugs, release updates based on user feedback	All Developers	Ongoing